Fields Institute Communications
Volume 00, 0000

On dynamic scheduling of a parallel server system with complete resource pooling

R. J. Williams
Department of Mathematics
University of California, San Diego
La Jolla CA 92093-0112, USA
williams@math.ucsd.edu

1 Introduction

We consider a dynamic scheduling problem for a parallel server queueing system. This system might be viewed as a model for a manufacturing or computer system, consisting of a bank of buffers for holding incoming jobs and a bank of servers for processing these jobs. Incoming jobs are classified into one of several different classes (or buffers). Jobs within a class are served on a first-in-first-out basis by one of a bank of parallel servers. Servers may have differing but overlapping capabilities and so may be able to service more than one class. The system manager seeks to minimize holding costs by dynamically allocating waiting jobs to available servers.

The parallel server system is described in more detail in Section 2 below. With the exception of a few special cases, the dynamic scheduling problem for this system cannot be analyzed exactly and it is natural to consider more tractable approximations. One class of such approximations are the so-called Brownian control problems, first introduced by Harrison in [**5**]. These are formal heavy traffic approximations to queueing control problems. Various authors (see for example [**3, 12, 13, 14, 19, 20, 23**]) have used analysis of these Brownian control problems, together with clever interpretation of their optimal (analytic) solutions to suggest "good" policies for the original queueing control problems.

For the parallel server system considered here, in a recent work, Harrison and López [**10**] studied the associated Brownian control problem and identified a condition under which the solution of that problem exhibits *complete resource pooling,* i.e., in the Brownian model, the efforts of the individual servers can be efficiently combined to act as a single pooled resource or "superserver". Under this condition, Harrison and López [**10**] conjecture that a "discrete review" scheduling policy (for the original parallel server system), obtained by using the BIGSTEP discretization procedure of Harrison [**7**], is asymptotically optimal in the heavy traffic limit.

1991 *Mathematics Subject Classification.* 60J70, 60K25, 60K30, 68M20, 90B35.

Research supported in part by NSF grant DMS 9703891. This paper is based on a talk given at the Fields Institute Workshop on Analysis and Simulation of Communication Networks held November 9–13, 1998.

Here, focussing on the parameter regime associated with the complete resource pooling condition of Harrison and López [**10**], we first review the formulation and solution of the Brownian control problem, and then we propose an alternative to the Harrison-López [**10**] discrete-review policy, namely, a simple dynamic threshold-type scheduling policy, which we conjecture is asymptotically optimal in the heavy traffic limit. Although our review of the Brownian control problem has many elements in common with the treatment in Harrison and López [**10**] and it certainly draws on the results proved there, our presentation differs from [**10**] in several respects. In particular, we have attempted to distill (and expand where necessary) results from prior works, especially [**11, 9, 10**]. The main novel aspects are the following: (a) we specify the properties of the primitive stochastic processes involved in our model for the parallel server system, and in terms of these we specify a cost function and define a notion of asymptotic optimality (a precise description of these items is needed for proofs of asymptotic optimality); (b) we motivate the notion of heavy traffic, as defined by Harrison [**9**] and Harrison and López [**10**], via an associated fluid model (this fluid model plays a role in proofs of asymptotic optimality, see e.g., [**1**]); (c) we describe how optimal controls for the Brownian control problem are derived from those for the *equivalent workload formulation* and emphasize the orthogonal decompositions of Euclidean space that play an essential role in this development (Harrison and López [**10**] cited prior general works of Harrison and Van Mieghem [**11**] and Harrison [**9**] for the reduction to the equivalent workload formulation, whereas here we make the connection explicit in the context of the parallel server problem); (d) our proposal for a dynamic threshold-type policy described in Section 6, and especially the server-buffer tree introduced there, are new. (For the example pictured in Figure 1, Marcel López (private communication) proposed certain activity and server trees. Although these trees were example specific and were not the same as the server-buffer tree described here, they provided the impetus to search for a general "tree" based policy. More recently, and independently of this work, Squillante, Xia, Yao and Zhang [**22**] have proposed the use of certain threshold-type policies tied to a tree structure for some parallel server examples arising in affinity scheduling of computer memory.) To support our conjecture that the proposed threshold policy is asymptotically optimal, we note that a proof of this for a two-server system is given in Bell-Williams [**1**]. Work in progress is directed to extending this to the many server case.

This paper is organized as follows. In Section 2, we describe the model of a parallel server system considered here. In Section 3, we review the notion of heavy traffic defined in [**9, 10**] using a linear program, and we interpret this in terms of the behavior of an associated fluid model. In Section 4, we describe the Brownian control problem. The cost function used there is an average discounted holding cost function. Following Harrison and López [**10**], in Section 5 we present the solution of a reduced form of this problem called the equivalent workload formulation [**11, 9**], under the complete resource pooling condition of [**10**]. This condition ensures that the Brownian model workload process is one-dimensional and from [**10**] we know this condition is equivalent to uniqueness of a solution to the dual to the heavy traffic linear program described in Section 3. In Section 6, we describe the dynamic threshold-type scheduling policy that we propose for the original parallel server problem.

Throughout the paper, vectors should be treated as column vectors unless indicated otherwise, inequalities between vectors should be interpreted componentwise,

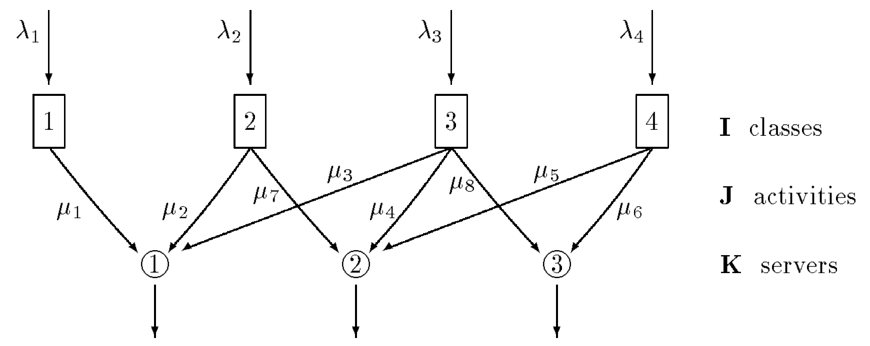

Figure 1 Example of a parallel server system.

the transpose of a vector a will be denoted by a', the diagonal matrix with the entries of a vector a on its diagonal will be denoted by $\text{diag}(a)$, and the dot product of two vectors a and b will be denoted by $a \cdot b$.

2 The Queueing Model: Parallel Server System

2.1 System structure. Our parallel server system consists of $\mathbf{I}$ infinite capacity buffers for holding jobs awaiting service, indexed by $i = 1, \ldots, \mathbf{I}$, and $\mathbf{K}$ (non-identical) servers working in parallel, indexed by $k = 1, \ldots, \mathbf{K}$ (see e.g., Figure 1). Each buffer has its own stream of jobs arriving from outside the system. Arrivals to buffer i are called class i jobs and jobs are ordered within the buffer according to their arrival times, with the earliest arrival being at the head of the line. Each job entering the system requires a single service before it exits the system. Several different servers may be capable of processing (or serving) a particular job class. Service of a given job class i by a given server k is called a processing activity. Although we shall not use this notation, such a processing activity can be associated with the ordered pair (i, k) and there is an upper bound of $\mathbf{I} \cdot \mathbf{K}$ on the number of processing activities. Due to system constraints, the actual number $\mathbf{J}$ of processing activities available may be less than this upper bound. In any event, we assume there are $\mathbf{J} \leq \mathbf{I} \cdot \mathbf{K}$ possible processing activities labelled by $j = 1, \ldots, \mathbf{J}$. The correspondences between activities and classes, and activities and servers, are described by two deterministic matrices C, A, where C is an $\mathbf{I} \times \mathbf{J}$ matrix with

$$C_{ij} = \begin{cases} 1 & \text{if activity } j \text{ processes class } i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

A is a $\mathbf{K} \times \mathbf{J}$ matrix with

$$A_{kj} = \begin{cases} 1 & \text{if server } k \text{ performs activity } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that each column of C contains the number one exactly once and similarly for A, since each activity j has exactly one class $i(j)$ and one server $k(j)$ associated with it. We also assume that each row of C and each row of A contains the number one at least once (i.e., each job class is capable of being processed by at least one activity and each server is capable of performing at least one activity).

For the example pictured in Figure 1, which is taken from the paper of Harrison and López [**10**], the matrices C and A are given by

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Once a job has commenced service at a server, it remains there until its service is complete, even if its service is interrupted for some time (e.g., by preemption by a job of another class). A server may not start on a new job of class i until it has finished serving any class i job that it is working on or that is in suspension at the server. In addition, a server cannot work unless it has a job to work on. When taking a new job from a buffer, a server always takes the job at the head-of-the-line. (For concreteness, we suppose that a deterministic tie-breaking rule is used when two (or more) servers want to simultaneously take jobs from the same buffer, e.g., there is an ordering of the servers and higher numbered servers take jobs before low numbered ones.) This setup allows a job to be allocated to a server just before it begins service, rather than upon arrival to the system. We assume that the system is initially empty.

2.2 Stochastic primitives. All random variables and stochastic processes used in our model descriptions are assumed to be defined on a given complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$. All continuous-time stochastic processes will be assumed to have paths that are right continuous with finite left limits. We shall frequently use the term process in place of stochastic process. The expectation operator under $\mathbb{P}$ will be denoted by $\mathbb{E}$. For $i = 1, 2, \ldots, \mathbf{I}$, we take as given a sequence of strictly positive i.i.d. random variables $\{u_i(\ell), \ell = 1, 2, \ldots\}$ with mean $\lambda_i^{-1} \in (0, \infty)$ and variance $a_i^2 \in [0, \infty)$. We interpret $u_i(\ell)$ as the interarrival time between the $(\ell - 1)^{st}$ and the ℓ^{th} arrival to class i where, by convention, the "0^{th} arrival" is assumed to occur at time zero. Setting $\xi_i(0) = 0$ and

$$\xi_i(n) = \sum_{\ell=1}^{n} u_i(\ell), \qquad n = 1, 2, \ldots, \tag{2.3}$$

we define

$$E_i(t) = \sup\{n \geq 0 : \xi_i(n) \leq t\} \qquad \text{for all } t \geq 0. \tag{2.4}$$

Then $E_i(t)$ is the number of arrivals to class i that have occurred in $[0, t]$, and λ_i is the long run arrival rate to class i. For $j = 1, 2, \ldots, \mathbf{J}$, we take as given a sequence of strictly positive i.i.d. random variables $\{v_j(\ell), \ell = 1, 2, \ldots\}$ with mean $\mu_j^{-1} \in (0, \infty)$ and variance $b_j^2 \in [0, \infty)$. We interpret $v_j(\ell)$ as the amount of service time required by the ℓ^{th} job processed by activity j, and μ_j is the long run rate at which activity j could process its associated class of job $i(j)$ if the associated server $k(j)$ worked continuously and exclusively on this class. For $j = 1, 2, \ldots, \mathbf{J}$, let $\eta_j(0) = 0$,

$$\eta_j(n) = \sum_{\ell=1}^{n} v_j(\ell), \qquad n = 1, 2, \ldots, \tag{2.5}$$

and

$$S_j(t) = \sup\{n \geq 0 : \eta_j(n) \leq t\} \qquad \text{for all } t \geq 0. \tag{2.6}$$

Then $S_j(t)$ is the number of jobs that activity j could process in $[0,t]$ if the associated server worked continuously and exclusively on the associated class of jobs during this time interval. The interarrival time sequences $\{u_i(\ell), \ell = 1,2,\ldots\}$, $i = 1,2,\ldots,\mathbf{I}$, and service time sequences $\{v_j(\ell), \ell = 1,2,\ldots\}$, $j = 1,2,\ldots,\mathbf{J}$, are assumed to be mutually independent.

The following exponential moment assumption ensures that certain *large deviation estimates* hold for the renewal processes E_i, $i = 1,\ldots,\mathbf{I}$, and $S_j, j = 1,\ldots,\mathbf{J}$. (We use this assumption in proving asymptotic optimality of certain threshold policies, cf. [**1**].)

Assumption 2.1 There is an open neighborhood $\mathcal{O}$ of $0 \in \mathbb{R}$ such that for all $s \in \mathcal{O}$,

$$\Lambda_{u_i}(s) \equiv \log \mathbb{E}[e^{su_i(1)}] < \infty \qquad \text{for} \quad i = 1,\ldots,\mathbf{I}, \tag{2.7}$$

and

$$\Lambda_{v_j}(s) \equiv \log \mathbb{E}[e^{sv_j(1)}] < \infty \qquad \text{for} \quad j = 1,\ldots,\mathbf{J}. \tag{2.8}$$

2.3 Scheduling control. Scheduling control is exerted through allocations of server time by each server to its associated processing activities. Our model structure allows dynamic sequencing and alternate routing of jobs. For example, if server k can perform more than one activity (i.e., $A_{kj} \neq 0$ for more than one j), then upon completing service of a job, server k can make a *sequencing decision*, i.e., which activity to perform next. This decision might depend for instance on how many jobs are in each of the buffers for the job classes that server k can process, or on other factors such as priorities for different job classes. In addition, if a given job class i can be processed by more than one activity (i.e., $C_{ij} \neq 0$ for more than one j), then jobs of class i may be serviced by one of a collection of servers and so simple *alternate routing* capabilities are encompassed here.

Formally, scheduling control is exerted by specifying a $\mathbf{J}$-dimensional stochastic process $T = \{T(t), t \geq 0\}$ where

$$T(t) = (T_1(t),\ldots,T_{\mathbf{J}}(t))' \qquad \text{for } t \geq 0, \tag{2.9}$$

and $T_j(t)$ is the cumulative amount of service time devoted to activity j by the associated server in the time interval $[0,t]$. Now T must satisfy certain properties that go along with its interpretation. Indeed, one could give a discrete-event type description of the properties that T must have, including any system specific constraints such as no preemption of service. However, for our analysis, we shall only need the properties of T described shortly. Let

$$I(t) = et - AT(t), \qquad t \geq 0, \tag{2.10}$$

where e is the $\mathbf{K}$-dimensional vector of all ones. Then for each $k = 1,\ldots,\mathbf{K}$, $I_k(t)$ is interpreted as the cumulative amount of time that server k has been idle up to time t. A natural constraint on T is that each component of the cumulative idletime process $I(\cdot)$ must be continuous and non-decreasing. This immediately implies the property that each component of T is Lipschitz continuous with Lipschitz constant less than or equal to one. For each $j = 1,\ldots,\mathbf{J}$, $S_j(T_j(t))$ is interpreted as the number of complete jobs processed by activity j in $[0,t]$. For $i = 1,\ldots,\mathbf{I}$, let

$$Q_i(t) = E_i(t) - \sum_{j=1}^{\mathbf{J}} C_{ij} S_j(T_j(t)), \tag{2.11}$$

which we write in vector form (with a slight abuse of notation for $S(T(t))$) as

$$Q(t) = E(t) - CS(T(t)). \tag{2.12}$$

Then $Q_i(t)$ is interpreted as the number of class i jobs that are either in queue or "in progress" (i.e., being served or in suspension) at time t.

We shall use the following minimal set of properties of any scheduling control T with associated queue length process Q and idletime process I. For all i, j, k,

$$T_j(t) \in \mathcal{F} \quad \text{for each } t \geq 0, \tag{2.13}$$

$$T_j(\cdot) \text{ is Lipschitz continuous (Lip. constant } \leq 1\text{),} \tag{2.14}$$
$$\text{non-decreasing and } T_j(0) = 0,$$

$$I_k(\cdot) \text{ is continuous, non-decreasing and } I_k(0) = 0, \tag{2.15}$$

$$Q_i(t) \geq 0 \text{ for all } t \geq 0. \tag{2.16}$$

For later reference, we collect here the queueing system equations for Q and I:

$$Q(t) = E(t) - CS(T(t)) \tag{2.17}$$

$$I(t) = et - AT(t) \tag{2.18}$$

where Q, T and I satisfy properties (2.13)–(2.16). We emphasize that these are descriptive equations satisfied by the queueing system, given C, A, E, S and a control T, which suffice for the purposes of our analysis. In particular, we do not intend them to be a complete discrete-event type description of the dynamics.

Remark The reader might expect that T should satisfy some additional non-anticipating property. Although this is a reasonable assumption to make, and indeed the policy we propose in Section 6 satisfies such a condition, we have not restricted T a priori in this way. Indeed, we conjecture that, for the parallel server system under the complete resource pooling condition, our policy is asymptotically optimal even when anticipating policies are allowed. This is related to the fact that the Brownian control problem has a so-called "pathwise solution", cf. [**10**].

The cost function we shall use involves linear holding costs associated with the expense of holding jobs of each class in the system until they have completed service. We defer the precise description of this cost function to a later section, since it is formulated in terms of normalized queue lengths, where the normalization is in diffusion scale.

Throughout this paper, we shall use the system pictured in Figure 2 with $\lambda_1 > \mu_1$ and $\mu_3 > \lambda_2$ as a simple two-server example of our parallel server system. This system is considered in more detail in [**1**] and was considered with particular parameter values, Poisson arrivals and deterministic service times in [**8**]. In our example, the arrival rate to buffer 1 exceeds the service rate of server 1 and the arrival rate to buffer 2 is less than the service rate of server 2, so we may consider server 2 as a helper to server 1. We further restrict the parameter region to that associated with heavy traffic, i.e., we restrict to the region in which the average excess load placed on buffer 1 is balanced by the average excess service capacity of server 2, i.e., $\frac{\lambda_1 - \mu_1}{\mu_2} = 1 - \frac{\lambda_2}{\mu_3}$. The next section describes how a heavy traffic parameter region may be identified for the *general* parallel server problem by considering an associated fluid model problem.

3 The Fluid Model and Heavy Traffic

Our model allows for alternate routing and so there is no conventional notion of heavy traffic, since the nominal (or average) load on a server depends on the

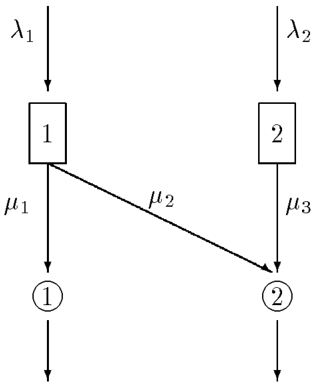

Figure 2 A two-server example.

routing policy. Harrison [**9**] (see also Laws [**19**] and Harrison-Van Mieghem [**11**]) has proposed a notion of heavy traffic for queueing networks with alternate routing and sequencing control. We attempt to motivate that definition here (in the context of our parallel server system) via desired behavior of an associated fluid model. This fluid model also plays a role in establishing asymptotic optimality of scheduling control policies.

Here we regard a flow as a continuous deterministic function. The *fluid model* corresponding to our parallel server queueing system is a formal deterministic (or law of large numbers) analogue in which the primitive stochastic processes E and S are replaced by linear flows which move at rates equal to the average rates of the original processes. In addition, the processes Q, I and T are replaced by flows $\bar{Q}$, $\bar{I}$ and $\bar{T}$ which must satisfy the *fluid model equations* (cf. (2.17)–(2.18)):

$$\bar{Q}(t) = \lambda t - R\bar{T}(t) \tag{3.1}$$

$$\bar{I}(t) = et - A\bar{T}(t) \tag{3.2}$$

where $R = C\mathrm{diag}(\mu)$ and for all i, j, k,

$$\bar{T}_j \text{ is Lipschitz continuous (Lip. constant } \leq 1\text{),} \tag{3.3}$$
$$\text{non-decreasing and } \bar{T}_j(0) = 0,$$
$$\bar{I}_k \text{ is continuous, non-decreasing and } \bar{I}_k(0) = 0, \tag{3.4}$$
$$\bar{Q}_i(t) \geq 0 \text{ for all } t \geq 0. \tag{3.5}$$

Equations (3.1)–(3.2) and (3.3)–(3.5) define the fluid model. A $\bar{T}$ satisfying (3.1)–(3.2) and (3.3)–(3.5) will be called a *fluid control.* One might interpret $\bar{T}$ as representing average (at a law of large numbers scale) or "nominal" allocations of service to the processing activities. For a given fluid control $\bar{T}$, we say the fluid system is *balanced* if the associated fluid "queue length" $\bar{Q}$ does not change with time (cf. Harrison [**6**]). Here, since the system starts empty, that means $\bar{Q} \equiv 0$. In addition, we say the fluid system *incurs no idleness* (or all fluid servers are fully occupied) if $\bar{I} \equiv 0$, i.e., $A\bar{T}(t) = et$ for all t.

Definition 3.1 The *fluid model* is in *heavy traffic* if the following two conditions hold:

(i) there is a unique fluid control $\bar{T}^*$ under which the fluid system is balanced, and

(ii) under $\bar{T}^*$, the fluid system incurs no idleness.

Remark The question of how to perform a heavy traffic analysis of a model when the above conditions are not satisfied is an interesting open problem. (For a start, see the related work by Laws [**18**].)

Since any fluid control is differentiable at almost every time (by (3.3)), we can convert the above notion of heavy traffic into one involving the rates $x^*(t) = \dot{\bar{T}}^*(t)$, where $\dot{}$ denotes time derivative. This leads to the notion of heavy traffic as formulated by Harrison [**9**] in terms of the following linear program.

Linear Program

$$\text{minimize } \rho \text{ subject to } Rx = \lambda, \quad Ax \leq \rho e \quad \text{and} \quad x \geq 0. \tag{3.6}$$

Remark Harrison [**9**] calls this the *static allocation problem.*

The following is Harrison's heavy traffic condition.

Assumption 3.2 There is a unique optimal solution (x^*, ρ^*) of the linear program (3.6). Moreover, that solution is such that $\rho^* = 1$ and $Ax^* = e$.

Lemma 3.3 *The fluid model is in heavy traffic if and only if Assumption 3.2 holds.*

Proof Suppose the fluid model is in heavy traffic. Then there is a fluid control $\bar{T}^*$ such that

$$0 = \lambda t - R\bar{T}^*(t) \qquad \text{and} \qquad 0 = et - A\bar{T}^*(t) \qquad \text{for all } \ t \geq 0. \tag{3.7}$$

Fix a $t > 0$ at which $\bar{T}^*$ is differentiable and let $x^* = \dot{\bar{T}}^*(t)$ there. Differentiating (3.7) we have

$$Rx^* = \lambda \qquad \text{and} \qquad Ax^* = e \tag{3.8}$$

and hence x^* is a feasible solution of the linear program (3.6) with $\rho = 1$. To show $(x^*, 1)$ is the unique optimal solution of the linear program, suppose there is another feasible solution (x, ρ) of the linear program with $\rho \leq 1$. Then, $\bar{T}(t) \equiv xt$ is a fluid control under which the fluid system is balanced and, by the uniqueness in the definition of heavy traffic, we must have $\bar{T} = \bar{T}^*$. Hence $x = x^*$ and $\rho = 1$.

Conversely, suppose there is a unique optimal solution (x^*, ρ^*) of the linear program (3.6), and $\rho^* = 1$, $Ax^* = e$. Let $\bar{T}^*(t) \equiv x^*t$ for all $t \geq 0$. Then under the fluid control $\bar{T}^*$, the fluid system is balanced and incurs no idleness. If there is another fluid control $\bar{T}$ under which the fluid system is balanced, then setting $x = \dot{\bar{T}}(t)$ for some t at which $\bar{T}$ is differentiable, we see that $x \geq 0$ (since $\dot{\bar{T}}(t) \geq 0$), $\lambda = Rx$ and $Ax \leq e$ (since $\dot{\bar{I}}(t) \geq 0$). Thus, $(x, 1)$ is a feasible solution of the linear program. By the uniqueness of the optimal solution, we must have $x = x^*$. It follows that $\dot{\bar{T}} = \dot{\bar{T}}^*$ a.e. and hence $\bar{T} \equiv \bar{T}^*$ since they both start from the same value. $\square$

We shall say that *our parallel server queueing system is in heavy traffic* if the associated fluid model is in heavy traffic. (In general, one may consider a sequence of queueing systems approaching heavy traffic. To simplify the exposition, here we focus on rescaling a single system and do not include the more general case, cf. [**1**].)

We make the following assumption henceforth.

Assumption 3.4 The parallel server system is in heavy traffic, i.e., Assumption 3.2 holds.

Example 3.5 For the two-server example pictured in Figure 2 with $\lambda_1 > \mu_1$, $\mu_3 > \lambda_2$, the heavy traffic condition reduces to $\frac{\lambda_1-\mu_1}{\mu_2} = 1 - \frac{\lambda_2}{\mu_3}$, and in this case the unique optimal solution of (3.6) has

$$x^* = \left(1, \frac{\lambda_1 - \mu_1}{\mu_2}, \frac{\lambda_2}{\mu_3}\right)'. \tag{3.9}$$

Example 3.6 As an example, Harrison and López **[10]** consider the system pictured in Figure 1 with

$$\lambda = \left(\frac{1}{2}, 1, 2, 3\right)', \qquad \mu = \left(\frac{5}{4}, \frac{5}{2}, 1, 2, 5, \frac{5}{2}, 4, \frac{4}{5}\right)'. \tag{3.10}$$

With these data, the heavy traffic condition is satisfied and the unique optimal solution of (3.6) has

$$x^* = \left(\frac{2}{5}, \frac{2}{5}, \frac{1}{5}, \frac{9}{10}, \frac{1}{10}, 1, 0, 0\right)'. \tag{3.11}$$

Example 3.7 Consider the system pictured in Figure 1 with

$$\lambda = (1, 1, 3, 1)', \qquad \mu = \left(\frac{5}{4}, 1, 5, 2, 4, 1, 4, 1\right)'. \tag{3.12}$$

With these data, the heavy traffic condition is satisfied and the unique optimal solution of (3.6) has

$$x^* = \left(\frac{4}{5}, 0, \frac{1}{5}, \frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}, 1\right). \tag{3.13}$$

For the formulation of the Brownian control problem, it will be helpful to distinguish basic activities j which have a strictly positive nominal fluid allocation level x_j^* from non-basic activities j for which $x_j^* = 0$. By relabelling the activities if necessary, we may and do assume henceforth that $x_j^* > 0$ for $j = 1, \ldots, \mathbf{B}$ and $x_j^* = 0$ for $j = \mathbf{B}+1, \ldots, \mathbf{J}$. Thus there are $\mathbf{B}$ basic activities and $\mathbf{J}-\mathbf{B}$ non-basic activities. Accordingly we partition the matrices R and A:

$$R = [H \quad M] \qquad \text{and} \qquad A = [B \quad N] \tag{3.14}$$

where H is $\mathbf{I}\times\mathbf{B}$, M is $\mathbf{I}\times(\mathbf{J}-\mathbf{B})$, B is $\mathbf{K}\times\mathbf{B}$ and N is $\mathbf{K}\times(\mathbf{J}-\mathbf{B})$.

4 Diffusion Scaling, Cost Function and Brownian Control Problem

Given the heavy traffic assumption of the previous section and the fluid model interpretation of this condition, to keep queue lengths from growing on average, it seems desirable to choose a control in the original parallel server system that on average allocates service to the processing activities in accordance with the proportions given by x^*. At first one might be tempted to set $T(t) = x^*t$ in the parallel server system. However, this is not a valid control for the system, since (2.16) will be violated due to the random fluctuations in the arrival process E and service process S. To see how to achieve the proportions x^* on average, and to do so in an optimal manner, we turn to a finer approximate model than the fluid model. This is called a Brownian model or Brownian control problem and it may be regarded as a formal diffusion approximation to the queueing control problem. Its relationship to the fluid model is analogous to the relationship between the central limit theorem and the law of large numbers.

To formally describe this Brownian model, we consider a sequence of parallel server systems indexed by r, where r tends to infinity through a sequence of values in $[1,\infty)$. These systems all have the same basic structure as that described in Section 2, except that the control policy and form of the cost function (which is defined below) are allowed to depend on r. Accordingly, we shall indicate the dependence of a control policy on r by appending a superscript to it: T^r. (A more complex setup can be treated, where one allows the distributions of the stochastic primitives to vary with r, cf. [**1**]. To simplify the exposition we have not included that generalization here.)

For a fixed r and control policy T^r, the associated queue length process $Q^r = (Q_1^r,\dots,Q_{\mathbf{I}}^r)'$ and idletime process $I^r = (I_1^r,\dots I_{\mathbf{K}}^r)'$ are given by (2.17)–(2.18) where the superscript r needs to be appended to Q, I and T there. The normalized queue length and idletime processes are defined by

$$\hat{Q}^r(t) = r^{-1}Q^r(r^2t), \qquad \hat{I}^r(t) = r^{-1}I^r(r^2t). \tag{4.1}$$

We consider an average cumulative discounted holding cost for the normalized queue length process and control T^r:

$$\hat{J}^r(T^r) = \mathbb{E}\left(\int_0^\infty e^{-\gamma t}\, h\cdot\hat{Q}^r(t)\,dt\right), \tag{4.2}$$

where $\gamma > 0$ is a fixed constant (discount factor) and $h = (h_1,\dots,h_{\mathbf{I}})'$, $h_i > 0$ for $i = 1,\dots,\mathbf{I}$, is a constant vector of holding costs per unit time per unit of normalized queue length. Recall that $\cdot$ denotes the dot product for two vectors.

To write equations for $\hat{Q}^r, \hat{I}^r$, we introduce centered and normalized versions $\hat{E}^r, \hat{S}^r$ of the primitive processes E, S:

$$\hat{E}^r(t) = r^{-1}\left(E(r^2t) - \lambda r^2 t\right), \qquad \hat{S}^r(t) = r^{-1}\left(S(r^2t) - \mu r^2 t\right), \tag{4.3}$$

a deviation process $\hat{Y}^r$ (which measures deviations from the nominal allocations given by x^*):

$$\hat{Y}^r(t) = r^{-1}\left(x^* r^2 t - T^r(r^2t)\right), \tag{4.4}$$

and

$$\hat{U}^r(t) = \begin{bmatrix} \hat{I}^r(t) \\ -\hat{Y}_N^r(t) \end{bmatrix}, \qquad K = \begin{bmatrix} B & N \\ 0 & -I \end{bmatrix}, \tag{4.5}$$

where $\hat{Y}_N^r$ denotes the $(\mathbf{J}-\mathbf{B})$-dimensional vector process consisting of the last $\mathbf{J}-\mathbf{B}$ components (the non-basic components) of $\hat{Y}^r$, which must be non-increasing since the corresponding components of x^* are zero. Also, I is the $(\mathbf{J}-\mathbf{B})\times(\mathbf{J}-\mathbf{B})$-identity matrix (not to be confused with the idletime process). Finally, we let

$$\bar{T}^r(t) = r^{-2}T^r(r^2t). \tag{4.6}$$

On substituting the above into (2.17)–(2.18) and using (3.8), we obtain

$$\begin{aligned} \hat{Q}^r(t) &= \hat{E}^r(t) - C\hat{S}^r(\bar{T}^r(t)) + R\hat{Y}^r(t), && (4.7)\\ \hat{U}^r(t) &= K\hat{Y}^r(t), && (4.8) \end{aligned}$$

where by (2.14)–(2.16) and the definition of non-basic activities, we have

$$\hat{U}^r(\cdot) \text{ is continuous, non-decreasing and } \hat{U}^r(0) = 0, \tag{4.9}$$

$$\hat{Q}_i^r(t) \ge 0 \text{ for all } t \ge 0 \text{ and } i \in \mathbf{I}. \tag{4.10}$$

Following the method proposed by Harrison [**5, 8, 10**], one arrives at the following formal Brownian control problem approximation (under diffusive scaling) to the control problem for the parallel server system. One can obtain this by formally passing to the limit as $r \to \infty$ in the control problem for the parallel server system. An important assumption in this formal procedure is that in fluid or law of large numbers scale, the allocation processes achieve the average levels $\bar{T}^*(t) \equiv x^* t$ for a balanced system in the heavy traffic limit, i.e., formally we have that as $r \to \infty$, $\bar{T}^r \Rightarrow \bar{T}^*$, where $\Rightarrow$ denotes convergence in distribution for processes (the topology on the path space for the processes is the usual Skorokhod $\mathbf{J}_1$-topology). The Brownian motion $\tilde{X}$ appearing in the Brownian control problem defined below is the formal limit in distribution of $\hat{E}^r(\cdot) - C\hat{S}^r(\bar{T}^r(\cdot))$, where functional central limit theorems for the independent renewal processes E, S and a time change theorem (together with the assumption that $\bar{T}^r \Rightarrow \bar{T}^*$), are used to derive the covariance matrix for this Brownian motion. The control process $\tilde{Y}$ in the Brownian control problem arises as a formal limit of the *deviation processes* $\hat{Y}^r$ which measure the deviation of the allocation processes T^r from the nominal fluid levels, i.e., formally $\hat{Y}^r \Rightarrow \tilde{Y}$ as $r \to \infty$.

Definition 4.1 (Brownian control problem)

$$\text{minimize } \mathbb{E}\left(\int_0^\infty e^{-\gamma t}\, h \cdot \tilde{Q}(t)\, dt\right) \tag{4.11}$$

using a $\mathbf{J}$-dimensional control process $\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_{\mathbf{J}})'$ such that

$$\tilde{Q}(t) \equiv \tilde{X}(t) + R\tilde{Y}(t) \quad \text{for all } t \geq 0, \tag{4.12}$$

$$\tilde{U}(t) = K\tilde{Y}(t) \quad \text{for all } t \geq 0, \tag{4.13}$$

$$\tilde{U}_k \text{ is non-decreasing and } \tilde{U}_k(0) = 0,\ k = 1, \ldots, \mathbf{K} + \mathbf{J} - \mathbf{B}, \tag{4.14}$$

$$\tilde{Q}_i(t) \geq 0 \text{ for all } t \geq 0,\ i = 1, \ldots, \mathbf{I}, \tag{4.15}$$

where $\tilde{X}$ is an $\mathbf{I}$-dimensional driftless Brownian motion that starts from the origin and has a diagonal covariance matrix whose i^{th} diagonal entry is equal to $\lambda_i^3 a_i^2 + \sum_{j:i(j)=i} \mu_j^3 b_j^2 x_j^*$.

5 Equivalent Workload Formulation

Following Harrison [**9**], Harrison and Van Mieghem [**11**] and Harrison and López [**10**], we seek to reduce the dimensionality of the Brownian control problem by projecting down to an "equivalent workload formulation". For this, let

$$\mathcal{N} = \{\delta \in \mathbb{R}^{\mathbf{I}} : \delta = Ry \text{ and } Ky = 0\}. \tag{5.1}$$

One may intuitively think of the elements of $\mathcal{N}$ as reversible displacements in the sense that in the Brownian model, when the "queue length" process $\tilde{Q}$ is strictly positive, one can adjust this queue length using a small displacement $\delta \in \mathcal{N}$ without incurring any additional "idleness" nor using any non-basic activities (i.e., without changing $\tilde{U}$). Since $\mathcal{N}$ is a vector space, such changes are "reversible". (For further discussion of the interpretation of $\mathcal{N}$, we refer the reader to Harrison [**9**].) In a sense, the idea of the equivalent workload formulation is to focus on controlling non-reversible displacements of "queue length", i.e., those in the orthogonal complement $\mathcal{N}^\perp$ of $\mathcal{N}$. The following theorem due to Harrison [**9**] is a key to this. The statement involves the dual to the linear program (3.6) introduced in Section 3.

Dual Program

$$\text{maximize } y \cdot \lambda \ \text{ subject to } \ y'R \leq z'A, \quad z \cdot e \leq 1 \ \text{ and } \ z \geq 0. \tag{5.2}$$

Theorem 5.1 (Harrison [9]) *Let $(y^1, z^1), \ldots, (y^{\mathbf{L}}, z^{\mathbf{L}})$ be the extreme points of the feasible set of solutions to the dual program (5.2) that satisfy $y^\ell \cdot \lambda = 1$, $\ell = 1, \ldots, \mathbf{L}$, (i.e., the maximum in the dual program (5.2) is achieved at these extreme points). Then,*

$$\mathcal{N}^\perp = span\{y^1, \ldots, y^{\mathbf{L}}\}. \tag{5.3}$$

We focus here on the case in which there is a unique optimal solution of the dual program (5.2) and hence $\mathcal{N}^\perp$ is one-dimensional. Theorem 5.3 below, due to Harrison and López [**10**], provides a convenient characterization of this case. For this, recall the definition of basic activities given at the end of Section 3. We shall also need the following notion of communicating servers.

Definition 5.2 Consider the graph $\mathcal{G}$ in which servers and buffers form the nodes and (undirected) edges between nodes are given by basic activities. We say that "all servers communicate via basic activities" if, for each pair of servers, there is a path in $\mathcal{G}$ joining the servers.

Theorem 5.3 (Harrison-López [10]) *The following conditions are equivalent:*

(i) *the dual program (5.2) has a unique optimal solution (y^*, z^*),*
(ii) *the number of basic activities $\mathbf{B}$ is equal to $\mathbf{I} + \mathbf{K} - 1$,*
(iii) *all servers communicate via basic activities.*

Remark Since $\lambda = Rx^*$ and $\lambda_i > 0$ for each i, each buffer i is connected to some server by some basic activity (i.e., there is an activity j such that $x_j^* > 0$ and $i(j) = i$). It follows that condition (iii) is equivalent to the condition that the entire graph $\mathcal{G}$ is connected. Indeed, we have the following interpretation of the conditions in Theorem 5.3.

Corollary 5.4 *The graph $\mathcal{G}$ is a tree if and only if the equivalent conditions of Theorem 5.3 hold.*

Proof It is well known that a graph with n nodes is a tree if and only if it is connected and has exactly $n-1$ edges (see for example, Berge [**2**], Theorem 3.1). Thus, since $\mathcal{G}$ has $\mathbf{I} + \mathbf{K}$ nodes, it is a tree if and only if the number of edges $\mathbf{B}$ is $\mathbf{I} + \mathbf{K} - 1$ (i.e., (ii) of Theorem 5.3 holds) and $\mathcal{G}$ is connected (i.e., (iii) of Theorem 5.3 holds). Since (ii) and (iii) of Theorem 5.3 are equivalent in our setting, the result follows. □

Henceforth we make the following assumption.

Assumption 5.5 All of the equivalent conditions (i)–(iii) of Theorem 5.3 hold.

The following three examples follow on from Examples 3.5, 3.6 and 3.7. Recall that we are assuming that the heavy traffic Assumption 3.4 holds.

Example 5.6 For Example 3.5, there is a unique solution of the dual program given by

$$y^* = \left(\frac{1}{\mu_1 + \mu_2}, \frac{\mu_2}{\mu_3(\mu_1 + \mu_2)}\right)', \qquad z^* = \left(\frac{\mu_1}{\mu_1 + \mu_2}, \frac{\mu_2}{\mu_1 + \mu_2}\right)'. \tag{5.4}$$

Example 5.7 For Example 3.6, considered by Harrison and López [**10**], there is a unique solution of the dual program given by

$$y^* = \left(\frac{1}{5}, \frac{1}{10}, \frac{1}{4}, \frac{1}{10}\right)', \qquad z^* = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)'. \tag{5.5}$$

Example 5.8 For Example 3.7, there is a unique solution of the dual program given by

$$y^* = \left(\frac{1}{2}, \frac{1}{16}, \frac{1}{8}, \frac{1}{16}\right)', \qquad z^* = \left(\frac{5}{8}, \frac{1}{4}, \frac{1}{8}\right)'. \tag{5.6}$$

Now, let (y^*, z^*) be the unique optimal solution of (5.2) and let g^* be the $(\mathbf{K}+\mathbf{J}-\mathbf{B})$-dimensional column vector where the first $\mathbf{K}$ components are given by the components of the vector z^* and the last $\mathbf{J}-\mathbf{B}$ components are given by the components of the (row) vector $(z^*)'N - (y^*)'M$. Thus, in symbols we have

$$g^* = ((z^*)', \;\; (z^*)'N - (y^*)'M)'. \tag{5.7}$$

Lemma 5.9 (Harrison-López [10]) *We have* $y^* > 0$, $z^* > 0$, $g^* \geq 0$ *and*

$$(y^*)'R = (g^*)'K. \tag{5.8}$$

Proof This result follows from Harrison-López [**10**] and is proved using the relation between the primal and dual linear programs. □

For $\tilde{Q}$ satisfying (4.12)–(4.15), define $\tilde{W} = y^* \cdot \tilde{Q}$, which Harrison [**9**] calls the (Brownian) workload. By Lemma 5.9 and (4.12)–(4.15),

$$\tilde{W}(t) = y^* \cdot \tilde{X}(t) + \tilde{V}(t) \quad \text{for all } t \geq 0, \tag{5.9}$$

where

$$\tilde{V} \equiv g^* \cdot \tilde{U} \text{ is non-decreasing and } \tilde{V}(0) = 0, \tag{5.10}$$

$$\tilde{W}(t) \geq 0 \text{ for all } t \geq 0. \tag{5.11}$$

Now, for each $t \geq 0$, since the holding cost vector $h > 0$ and $y^* > 0$, we have

$$h \cdot \tilde{Q}(t) = \sum_{i=1}^{\mathbf{I}} \left(\frac{h_i}{y_i^*}\right) y_i^* \tilde{Q}_i(t) \geq c^* \tilde{W}(t) \tag{5.12}$$

where

$$c^* \equiv \min_{i=1}^{\mathbf{I}} \left(\frac{h_i}{y_i^*}\right). \tag{5.13}$$

Thus, for each t, we might seek to minimize $h \cdot \tilde{Q}(t)$ by minimizing $\tilde{W}(t)$ and achieving equality in (5.12). Now, it is straightforward to see that any solution pair $(\tilde{W}, \tilde{V})$ of (5.9)–(5.11) must satisfy

$$\tilde{V}(t) \geq \tilde{V}^*(t) \equiv \sup_{0 \leq s \leq t} \left(-y^* \cdot \tilde{X}(s)\right), \tag{5.14}$$

and hence $\tilde{W}(t) \geq \tilde{W}^*(t)$ where

$$\tilde{W}^*(t) \;\; = \;\; y^* \cdot \tilde{X}(t) + \tilde{V}^*(t). \tag{5.15}$$

The process $\tilde{W}^*$ is the one-dimensional *reflected Brownian motion* driven by the one-dimensional Brownian motion $y^* \cdot \tilde{X}$, and $\tilde{V}^*$ is its local time at zero (see e.g., [**4**], Chapter 8). In particular, $\tilde{V}^*$ can have a point of increase at time t only if $\tilde{W}^*(t) = 0$.

Now, let i^* be a class index such that $c^* = h_{i^*}/y^*_{i^*}$, i.e., the minimum in (5.13) is achieved at $i = i^*$, and let k^* be a server that can serve class i^* via a basic activity. Then the following choices $\tilde{Q}^*$ and $\tilde{U}^*$ for $\tilde{Q}$ and $\tilde{U}$ ensure that for each

$t \geq 0$, properties (4.14)–(4.15) hold and the inequality in (5.12) is an equality with $\tilde{W}(t) = \tilde{W}^*(t)$ there:

$$\tilde{Q}^*_{i^*}(t) = \tilde{W}^*(t)/y^*_{i^*}, \qquad \tilde{Q}^*_i(t) = 0 \ \text{ for all } i \neq i^*, \tag{5.16}$$
$$\tilde{U}^*_{k^*}(t) = \tilde{V}^*(t)/z^*_{k^*}, \qquad \tilde{U}^*_k(t) = 0 \ \text{ for } k \neq k^*. \tag{5.17}$$

By a general result of Harrison and Van Mieghem [**11**] and Harrison [**9**], it is known that one can find control processes $\tilde{Y}^*$ such that (4.12)–(4.15) hold with $\tilde{Q}^*, \tilde{Y}^*, \tilde{U}^*$ in place of $\tilde{Q}, \tilde{Y}, \tilde{U}$ there. Here we indicate how such controls $\tilde{Y}^*$ are derived from $\tilde{Q}^*, \tilde{U}^*$. For this we need to elaborate on the role of the space $\mathcal{N}$. We first note that $Ky = 0$ implies that the non-basic components of y must all be zero. It follows from this and the decompositions of the matrices R and A (cf. (3.14)), that

$$\mathcal{N} = \{Hy : By = 0, \ y \in \mathbb{R}^{\mathbf{B}}\}, \tag{5.18}$$

i.e., if we let $\mathcal{B}$ denote the null space of B in $\mathbb{R}^{\mathbf{B}}$, then $\mathcal{N}$ is the image of $\mathcal{B}$ under H. Since $Ax^* = e$, each server must be serviced by at least one basic activity, and hence the linear mapping B is onto $\mathbb{R}^{\mathbf{K}}$. Let $B^\dagger$ denote the right inverse of B whose domain is $\mathbb{R}^{\mathbf{K}}$ and whose range is $\mathcal{B}^\perp$. In particular, we have $BB^\dagger$ is equal to the identity mapping on $\mathbb{R}^{\mathbf{K}}$. Since the range of B is $\mathbf{K}$-dimensional, it follows under Assumption 5.5 that the null space $\mathcal{B}$ of B is $\mathbf{B} - \mathbf{K} = (\mathbf{I} - 1)$-dimensional. Now, by our assumptions, $\mathcal{N}^\perp$ is one-dimensional and hence $\mathcal{N}$ is $(\mathbf{I} - 1)$-dimensional. Since this equals the dimension of $\mathcal{B}$, it follows that H is invertible as a mapping from $\mathcal{B}$ to $\mathcal{N}$. This inverse can be extended as a linear map defined on all of $\mathbb{R}^{\mathbf{I}} = \mathcal{N} \oplus \mathcal{N}^\perp$ by defining it to be zero on $\mathcal{N}^\perp$. We denote this extended mapping by H^*.

Now we are ready to define $\tilde{Y}^*$. Firstly, we set the non-basic components $\tilde{Y}^*_N$ to be identically zero and so it remains only to determine the basic components $\tilde{Y}^*_B$. Let $\tilde{I}^*$ be the $\mathbf{K}$-dimensional process given by the first $\mathbf{K}$ components of $\tilde{U}^*$. Eliminating the non-basic components from (4.12)–(4.15) and using the decompositions of R, A (cf. (3.14)), we see that it suffices to find a $\mathbf{B}$-dimensional process $\tilde{Y}^*_B$ such that

$$\tilde{Q}^* = \tilde{X} + H\tilde{Y}^*_B \tag{5.19}$$
$$\tilde{I}^* = B\tilde{Y}^*_B. \tag{5.20}$$

Using the decomposition $\mathbb{R}^{\mathbf{B}} = \mathcal{B} \oplus \mathcal{B}^\perp$ and recalling that the range of the right inverse $B^\dagger$ is $\mathcal{B}^\perp$, we see that any solution of (5.20) must be of the form

$$\tilde{Y}^*_B = B^\dagger \tilde{I}^* + \Lambda, \qquad \Lambda \in \mathcal{B}. \tag{5.21}$$

For such a $\tilde{Y}^*_B$ to also satisfy (5.19), we must have

$$\tilde{Q}^* - \tilde{X} - HB^\dagger \tilde{I}^* = H\Lambda. \tag{5.22}$$

Since H is invertible as a map from $\mathcal{B}$ onto $\mathcal{N}$, this equation has a (unique) solution $\Lambda \in \mathcal{B}$ if and only if the left member above is in $\mathcal{N}$. Since $\mathcal{N}^\perp$ is generated by y^* and $(y^*)'H = (z^*)'B$ (by (5.8)), using the definitions of $\tilde{W}^*, \tilde{U}^*, \tilde{V}^*$, we have

$$y^* \cdot (\tilde{Q}^* - \tilde{X} - HB^\dagger \tilde{I}^*) = \tilde{W}^* - y^* \cdot \tilde{X} - (z^*)'BB^\dagger \tilde{I}^* = \tilde{W}^* - y^* \cdot \tilde{X} - \tilde{V}^* = 0. \tag{5.23}$$

Thus there is a unique solution $\Lambda \in \mathcal{B}$ of (5.22) given by $\Lambda = H^*(\tilde{Q}^* - \tilde{X} - HB^\dagger \tilde{I}^*)$ and so we set

$$\tilde{Y}^*_B = B^\dagger \tilde{I}^* + H^*(\tilde{Q}^* - \tilde{X} - HB^\dagger \tilde{I}^*). \tag{5.24}$$

It can be readily verified that this is an optimal solution for the Brownian control problem (cf. [**10**]) and the associated minimum cost is

$$J^* \equiv \mathbb{E}\left(\int_0^\infty e^{-\gamma t}\, h \cdot \tilde{Q}^*(t)\, dt\right) = c^*\mathbb{E}\left(\int_0^\infty e^{-\gamma t}\, \tilde{W}^*(t)\, dt\right). \tag{5.25}$$

Now, even though the Brownian control problem can be analyzed exactly (as above), its solution does not automatically translate to a policy in the original parallel server system. However, some desirable features are suggested by the form of (5.16)–(5.17), namely,

(a) try to keep the bulk of the work in the class i^* with the lowest (or equal lowest) ratio of cost to workload contribution, i.e., the class with the lowest value of h_i/y_i^*,
(b) try to ensure that the bulk of the idleness is incurred only when there is no work in the entire system, and
(c) try to ensure that the bulk of the idletime is incurred by server k^* alone.

Harrison [**7**] has proposed a general scheme (called BIGSTEP) for obtaining candidate policies for the original queueing control problem from the solution of the associated Brownian control problem. The policies obtained in this manner are so-called discrete-review policies which allow review of the system status and changes in the control policy only at a fixed discrete set of times. In a sense, this scheme is a discretization of the problem of inferring control policies for the queueing model from the solution of the associated Brownian control problem. For the two-server example pictured in Figure 2 (and with Poisson arrivals, deterministic service times and particular values for $\lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3$), a family of discrete-review policies was constructed and shown to be asymptotically optimal in Harrison [**8**]. Based on their solution of the Brownian control problem and the general scheme laid out by Harrison [**7**], Harrison and López [**10**] proposed use of a family of discrete review policies for the many server problem, but did not prove asymptotic optimality of such a family.

Another approach to translation of solutions of Brownian control problems into viable policies has been proposed by Kushner et al. (see e.g., [**15, 16, 17**]), however this also involves discretization by way of numerical approximation. We note that of the works by Kushner et al. mentioned above, the paper by Kushner and Chen [**15**] is the closest to the current one in that it considers a parallel server model. However, it is in a very different parameter regime, namely one that corresponds to heavy traffic but with *no resource pooling*.

Assuming the complete resource pooling condition, in the next section we describe a simple "continuous review" policy for the parallel server system, which allows changes in the control to be made at random times and in particular at times when the system status changes. Our policy is a dynamic priority policy in which priorities for certain "transition" activities depend on the number of jobs in the associated class relative to certain threshold or "safety-stock" levels. Changes in the priorities only occur as a threshold is crossed. We conjecture that the policy proposed is asymptotically optimal. In [**1**], we have already shown that this is so for the two parallel server case described earlier. In work in progress, this case is being used as a stepping stone to a treatment of the many server case.

6 Threshold Policy and Asymptotic Optimality

In this section, we describe a dynamic threshold policy which we propose as a candidate for an asymptotically optimal control policy in our parallel server system. (A notion of asymptotic optimality is described formally at the end of this section). We note that there can be many asymptotically optimal policies. We simply propose the following as one that is intuitively appealing and that is easy to describe. Variations on this policy are certainly possible. For example, to reduce "chattering" back and forth across a single threshold, one might introduce a second threshold and an associated hysteretic type of strategy (cf. [**21**]).

In reviewing the description of our threshold policy, it may help readers to keep the following intuitive characteristics in mind. Our policy is aimed at achieving the desirable properties (a)–(c) listed in the previous section. The key to the description of this policy is a decomposition of the server-buffer tree $\mathcal{G}$ identified in Corollary 5.4 and an associated protocol for the dynamic allocation of class priorities at each server. This protocol is described in an iterative manner, working from the bottom of the tree up towards the root. A server tree $\mathcal{S}$, which results from suppressing the buffers in $\mathcal{G}$, is helpful in describing this iterative procedure. (In this paper, each tree with a distinguished root will be assumed to grow downwards from its root and so the root will be at the highest level.) The server k^* at the root is one that can service the "cheapest" class i^*. A mechanism for choosing this server is described as part of the policy. The idea behind our policy is to keep servers below the root level busy the bulk of the time (indeed, they should only be rarely idle and their idletime should vanish on diffusion scale as the heavy traffic limit is approached), while simultaneously preventing the queue lengths of the classes associated with those servers (except i^*) from growing appreciably on diffusion scale. Transition buffers which link one level of servers to those at the next highest level are used to achieve these two competing goals. We call the classes associated with these buffers transition classes. In brief, when the queue length for a transition class gets below a threshold, any service of that class by the server higher up the tree is suspended and this causes temporary overloading (on average) of the servers below, which prevents these servers from incurring much idleness. When the queue length for the transition class builds up to a level above its threshold, then assistance from the higher server is reactivated and the servers below are temporarily underloaded (on average) and so queue lengths for classes serviced by these servers are prevented from growing too large. The effect of this policy is to allow the movement (via the transition buffers) of excess work from lower level to higher level buffers (and eventually by an upwards cascade to the buffer for class i^*), while simultaneously keeping all servers busy the bulk of the time, unless the entire system is empty and even then to ensure that the vast majority of the idleness is incurred by the server at the root of the tree.

Recall that we are assuming throughout that the Assumptions 3.4 of heavy traffic and 5.5 of complete resource pooling hold. In the following, when describing the server-buffer tree $\mathcal{G}$, the terms class and buffer will be used interchangeably. In reviewing the description of the following threshold policy, the reader may find it helpful to refer to the examples which follow it. (The second and third examples both pertain to the model in Figure 1. However, they have different parameters and the server-buffer and server trees have different shapes in the two cases. In particular, for the third example, the server tree is not linear.)

Threshold Policy

(i) *Decomposition of the server-buffer tree.*
The first task is to associate exactly one server with each class, where the server associated with a class is linked to the class by a basic activity. Now each class is serviced by at least one basic activity because $\lambda = Rx^*$ and $\lambda_i > 0$ for all i. Thus, for each class i, we may and do choose a basic activity j such that $i(j) = i$. (This choice is not necessarily unique.) The server *associated* with class i is then $k(j)$. Turning this association around, we see that each server has a collection (possibly empty) of classes associated with it and there is no overlap of the collections associated with different servers, since no class is *associated* with more than one server. Moreover, each class belongs to one of the "server collections". (One may think of this as providing a partition of the $\mathbf{I}$ classes into $\mathbf{K}$ collections, one for each server, where some of the collections may be empty.) We represent each collection and the associated server by a simple tree with at most one level below the root. For this, the server is placed at the root and there is one edge (basic activity) leading from the root to each class associated with that server. (It is sometimes helpful to imagine that each simple tree can be "folded up" into its root server node to "hide" the classes associated with the server. On the other hand, the server node can be "opened up" to reveal the classes when needed.) Note that these simple trees are isolated from one another because no class is associated with more than one server. Now the total number of edges in this forest of $\mathbf{K}$ "simple trees" is equal to the number of classes $\mathbf{I}$. By property (ii) of Theorem 5.3, that leaves $\mathbf{K}-1$ edges representing basic activities that have not been "used" in this forest of simple trees. We know from property (iii) of Theorem 5.3 that each server communicates with every other server via basic activities. That is, if we now add in the edges (think of them as being dotted) corresponding to basic activities that have not yet been used in the forest of simple trees, the resulting graph $\mathcal{G}$ (which is the same as the server-buffer graph described in Corollary 5.4) is connected. In fact, we know from Corollary 5.4 that the graph is a tree. Furthermore, if we view the $\mathbf{K}$ simple trees as nodes (in their folded up configuration) in a connected graph with $\mathbf{K}-1$ dotted edges between the nodes, then this graph is a tree (cf. Corollary 5.4). We call this graph the server tree $\mathcal{S}$. Examples of forests of simple trees (linked by dotted edges) associated with Examples 3.5, 3.6, 3.7 are given in Figures 3, 5, 7 below.

(ii) *Root.*
Recall the class i^* chosen to achieve the minimum in (5.13). We can and do choose k^* to be the server *associated* with this class via the procedure described in (i) above. Then we take the server k^* to be at the root of the server tree $\mathcal{S}$ and at the root of the server-buffer tree $\mathcal{G}$. One may now think of placing the root at the highest level and letting the servers and buffers in $\mathcal{G}$ (or just the servers in the server tree) cascade below it in a hierarchy of alternating levels of servers and buffers where lower levels are further from the root. (This realignment of the server-buffer tree may change the original ordering within the simple trees of (i), i.e., it may no longer be true that all buffers lie below their *associated* servers.)

(iii) *Transition buffers and transition activities.*
Our proposed policy will only make use of basic activities. Among these,

so-called transition activities will play a key role. We now define these transition activities and transition buffers (and classes). Consider a dotted edge as described in (i). In the server-buffer graph $\mathcal{G}$, one end of the dotted edge is a buffer and the other is a server. We call the buffer attached to such a dotted edge a *transition buffer.* The corresponding class is called a transition class. Each transition buffer has exactly one basic activity linking it to a server higher up the tree (i.e., closer to the root). (There is at least one such basic activity because the graph $\mathcal{G}$ is connected. On the other hand, there is at most one such activity because if there were more than one, then there would be more than one path from the transition buffer to the root and this would contradict the fact that the graph $\mathcal{G}$ is a tree.) We call the basic activity that directly links a transition buffer to a server higher up the tree a *transition activity.* We note in passing that there can be more than one basic activity leading from a transition buffer to servers *lower* down the tree $\mathcal{G}$. One might like to think of edges representing transition activities as being colored red (labelled with a T in Figures 4, 6, 8). With the exception of the root, each server is linked directly by a basic activity to exactly one transition class that is at the next level up the server-buffer tree $\mathcal{G}$. We call this the *transition class* for the server. The tree structure of the graph $\mathcal{G}$ ensures that there is only one such class for each server, although two or more servers may have the same transition class.

Via the association chosen in (i), each transition buffer is associated either with the server immediately above it in the tree $\mathcal{G}$ or with one of the servers immediately below it. To assist with visualization of our control policy, now that a hierarchy (given by distance from the root) has been established in the server-buffer tree $\mathcal{G}$, it will be convenient to redefine the *association* of (i) so that transition buffers are always associated with the servers that lie immediately above them in the tree. There is no loss in doing this, since the resulting server tree and server-buffer tree are the same, it is just that the partitioning into simple trees may be slightly different. Indeed, with this reassociation, the image in (i) of buffers *associated* with servers being situated below the servers is restored in the tree $\mathcal{G}$.

(iv) *The policy.*

To see how to implement our threshold priority policy, we focus firstly on the server tree $\mathcal{S}$ and imagine it arranged in levels with the root at the highest level. We begin by focussing on the servers at the lowest level. Now consider each of these servers within the server-buffer tree $\mathcal{G}$. Each server should service their *associated* classes (where the association is via the modification of the server-buffer tree $\mathcal{G}$ described at the end of (iii)) and their transition class according to a priority scheme that gives lowest priority to the transition class (the class that is serviced by that server and the next server up the tree). The priority ranking of the classes at the lowest level, other than the transition class, is not so important. For instance, one might rank the classes so that for a given server, the higher numbered classes receive priority over the lower numbered ones. For future use, we place a threshold on the transition class for each of the lowest level servers, such that when the number of jobs in the class gets below the threshold, any service of that class by the server at the next highest level up the tree will be suspended. That is, the activity associated with the "red" edge leading from the transition class will be suspended. (We conjecture that a threshold of size $c \log r$ for

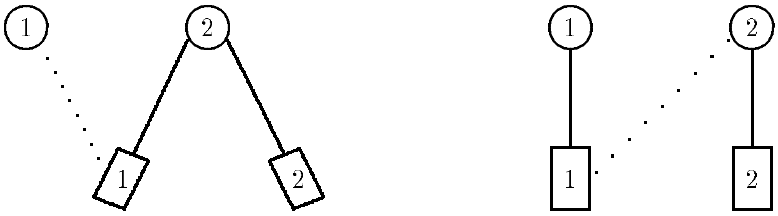

Figure 3 Two possible simple tree decompositions for the two server network pictured in Figure 2. Edges joining simple trees together are shown as dotted.

the r^{th} system, with c sufficiently large, but independent of r, will suffice for a proof of asymptotic optimality. This form for the threshold is motivated by our anticipation that a proof will use large deviation estimates to show that idleness is only rarely incurred at any server but the root server.)

Now go to the next level up the server tree $\mathcal{S}$. This level may have terminal server nodes and server nodes that lead to server nodes lower down the server tree. Let each server at this level perform its basic activities in a prioritized manner such that, in the server-buffer tree $\mathcal{G}$, activities leading (via transition buffers) to server nodes lower down the tree are given highest priority (if there is more than one such activity, one can for example rank them so that activities serving higher numbered classes are served first). For this, activities in suspension, due to the number of jobs in a transition class being below its threshold, are not performed. Next priority is given to basic activities that service classes that are only served by that server, and lowest priority is given to the activity serving the transition class to the next highest level. This transition class should again have a threshold placed on it such that service of that class by the server at the next highest level is suspended when the number of jobs in the class gets below the threshold level.

This process is repeated until the root of the server tree is reached. At the root, the same procedure is applied as for lower levels in the tree, except that there are no activities above the root server in the server-buffer tree and an overriding rule is that lowest priority is given to the "cheapest" class i^*.

Remark Although the above policy allows for preemption of service, there is a corresponding policy without preemption that we conjecture has the same behavior in the heavy traffic limit, since in that regime a maximum of $\mathbf{J}$ jobs (in suspension or not) should not impact the asymptotic behavior of the system.

Example 6.1 Consider the example pictured in Figure 2 with the data described in Example 3.5, and suppose that $\frac{h_1}{y_1^*} > \frac{h_2}{y_2^*}$, so that $i^* = 2$ and $k^* = 2$. The two possible simple tree decompositions are pictured in Figure 3 (Edges joining the simple trees together are shown dotted). Both of these decompositions lead to the same server tree and the same server-buffer tree which are both pictured in Figure 4.

Example 6.2 Consider the example pictured in Figure 1 with the data described in Example 3.6, and suppose that $h = \left(1, \frac{3}{5}, 2, 1\right)'$. Then $c^* = 5$, $i^* = 1$ and

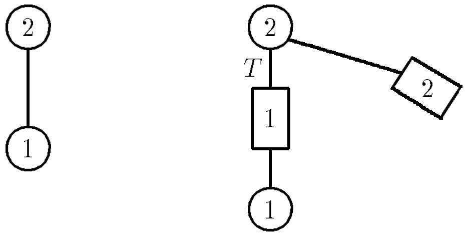

Figure 4 The server tree (on the left) and server-buffer tree (on the right) for the two server network of Figure 2 with $h_1/y_1^* > h_2/y_2^*$. Dotted edges have now been made solid. The "red" edge representing a transition activity has a T beside it.

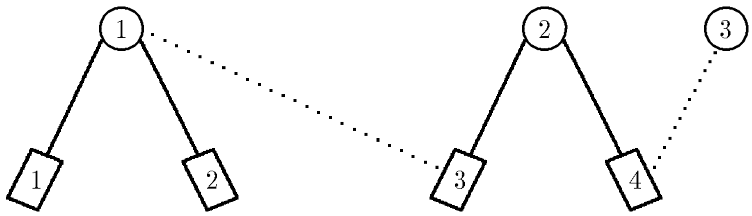

Figure 5 One possible simple tree decomposition for the three-server network pictured in Figure 1 with the data of Example 6.2. Dotted edges link simple trees together.

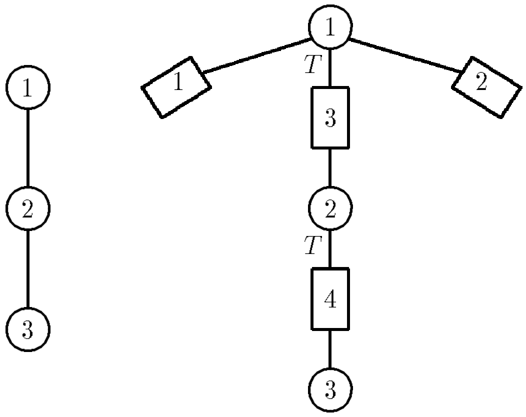

Figure 6 The server tree (on the left) and server-buffer tree (on the right) for the three-server network of Figure 1 with the data of Example 6.2 and simple tree decomposition of Figure 5. Dotted edges have now been made solid and "red" edges representing transition activities have a T beside them.

$k^* = 1$, cf. [**10**]. A possible simple tree decomposition of this network is pictured in Figure 5. The resulting server tree and server-buffer tree is pictured in Figure 6. The simple tree decomposition is not unique for this example, but the server tree

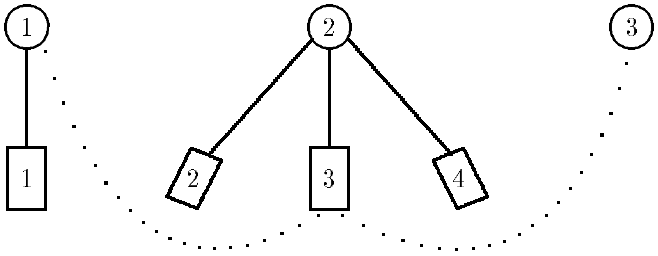

Figure 7 One possible simple tree decomposition for the three-server network pictured in Figure 1 with the data of Example 6.3. Dotted edges connect simple trees.

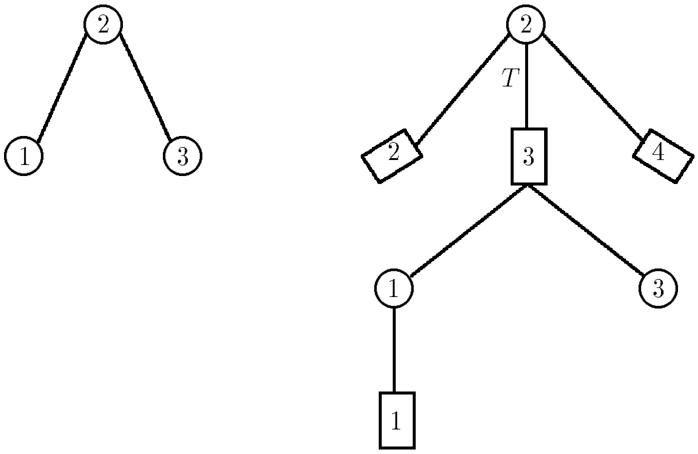

Figure 8 A server tree (on the left) and server-buffer tree (on the right) for the three-server network pictured in Figure 1 with the data of Example 6.3 and simple tree decomposition of Figure 7. Dotted edges have now been made solid and the "red" edge representing a transition activity has a T beside it.

and server-buffer tree are unique (since there is only one possible choice for k^* with the given data).

Example 6.3 Consider the example pictured in Figure 1 with the data of Example 3.7, and suppose that $h = (3, 1, \frac{1}{2}, 1)$. Then $c^* = 4$ and $i^* = 3$. A possible simple tree decomposition of this network is pictured in Figure 7. Given this choice of decomposition, we choose $k^* = 2$. The resulting server tree and server-buffer tree is pictured in Figure 8. Neither the simple tree decomposition, nor the server-buffer tree is unique for this example.

Recall the definition (4.2) of the cost $\hat{J}^r(T^r)$ associated with the use of control T^r in the r^{th} system, and of the optimal cost J^* for the Brownian control problem, which is given by (5.25). If we let $T^{r,*}$ denote service allocation processes associated with implementation of the threshold policy described above, then we conjecture that the policy is asymptotically optimal in the sense that if $\{T^r\}$ is any other sequence of scheduling control policies, then

$$\liminf_{r\to\infty} \hat{J}^r(T^r) \geq J^* = \lim_{r\to\infty} \hat{J}^r(T^{r,*}). \tag{6.1}$$

That is, the proposed policy achieves the optimal cost J^* of the Brownian control problem in the heavy traffic limit and this is the best that one can do asymptotically. This result is established in [**1**] for the two server system pictured in Figure 2.

Acknowledgements The author is grateful to Marcel López and Michael Harrison for access to a preliminary version of their paper [**10**], for related discussions which provided the inspiration for the work described here and for helpful comments on a preliminary version of this paper.

References

[1] Bell, S. L., and Williams, R. J. *Dynamic scheduling of a system with two parallel servers in heavy traffic with complete resource pooling: asymptotic optimality of a continuous review threshold policy*, preprint 1999, pp. 1–36.

[2] Berge, C. *Graphs*, North Holland, Amsterdam, 1985.

[3] Chevalier, P. B., and Wein, L. *Scheduling networks of queues: heavy traffic analysis of a multistation closed network*, Operations Res. **41** (1993), 743–758.

[4] Chung, K. L., and Williams, R. J. *Introduction to Stochastic Integration*, 2nd edition, Birkhäuser, Boston, 1990.

[5] Harrison, J. M. *Brownian models of queueing networks with heterogeneous customer populations*, in "Stochastic Differential Systems, Stochastic Control Theory and Their Applications", IMA Volume 10, W. Fleming and P.L. Lions (eds.), Springer Verlag, New York, 1988, pp. 147–186.

[6] Harrison, J. M. *Balanced fluid models of multiclass queueing networks: a heavy traffic conjecture,* in "Stochastic Networks", F. P. Kelly and R. J. Williams (eds.), Springer-Verlag, 1995, pp. 1–20.

[7] Harrison, J. M. *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.), Oxford University Press, 1996, pp. 57–90.

[8] Harrison, J.M. *Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies*, Ann. Appl. Prob. **8** (1998), 822–848.

[9] Harrison, J. M. *Brownian models of open processing networks: canonical representation of workload*, to appear in Ann. Appl. Prob.

[10] Harrison, J. M., and López, M. J. *Heavy traffic resource pooling in parallel-server systems*, to appear in Queueing Systems.

[11] Harrison, J. M., and Van Mieghem, J. A. *Dynamic control of Brownian networks: state space collapse and equivalent workload formulations*, Ann. Appl. Prob. **7** (1997), 747–771.

[12] Harrison, J. M., and Wein, L. *Scheduling networks of queues: heavy traffic analysis of a simple open network*, Queueing Systems **5** (1989), 265–280.

[13] Harrison, J. M., and Wein, L. *Scheduling networks of queues: heavy traffic analysis of a two-station closed network*, Operations Res. **38** (1990), 1052–1064.

[14] Kelly, F. P., and Laws, C. N. *Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling*, Queueing Systems **13** (1993), 47–86.

[15] Kushner, H. J., and Chen, Y.N. *Optimal control of assignment of jobs to processors under heavy traffic*, to appear in *Stochastics.*

[16] Kushner, H. J., and Dupuis, P. *Numerical Methods for Stochastic Control Problems in Continuous Time*, Springer-Verlag, New York, 1992.

[17] Kushner, H. J., and Martins, L. F. *Numerical methods for stochastic singular control problems*, SIAM J. Control and Optimization **29** (1991), 1443–1475.

[18] Laws, C. N. *Dynamic Routing in Queueing Networks*, Ph.D. dissertation, Statistical Laboratory, University of Cambridge, U.K., 1990.

[19] Laws, C. N. *Resource pooling in queueing networks with dynamic routing*, Adv. Appl. Prob. **24** (1992), 699–726.

[20] Laws, C. N., and Louth, G. M. *Dynamic scheduling of a four-station queueing network*, Prob. Eng. Inf. Sci. **4** (1990), 131–156.

[21] Roughan, M., and Pearce, C. E. M. *A martingale analysis of hysteretic overload control*, preprint, 1999.

[22] Squillante, M., Xia, C. H., Yao, D., and Zhang, L. *Optimal control of parallel-server fluid networks — with applications in affinity scheduling*, presentation at the 10th Informs Applied Probability Conference, Ulm, Germany, July 26–28, 1999.

[23] Wein, L. *Scheduling networks of queues: heavy traffic analysis of a two-station network with controllable inputs*, Operations Res. **38** (1990), 1065–1078.