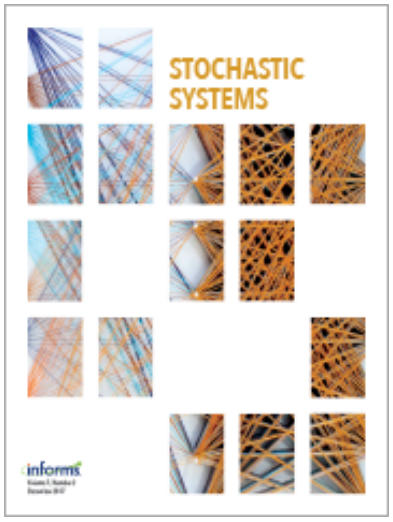

Stochastic Systems

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Resource Sharing Networks: Overview and an Open Problem

J. Michael Harrison, Chinmoy Mandayam, Devavrat Shah, Yang Yang

To cite this article:
J. Michael Harrison, Chinmoy Mandayam, Devavrat Shah, Yang Yang (2014) Resource Sharing Networks: Overview and an Open Problem. Stochastic Systems 4(2):524-555. https://doi.org/10.1287/13-SSY130

Stochastic Systems
2014, Vol. 4, No. 2, 524–555
DOI: 10.1214/13-SSY130

RESOURCE SHARING NETWORKS: OVERVIEW AND AN OPEN PROBLEM

By J. Michael Harrison*, Chinmoy Mandayam*, Devavrat Shah† and Yang Yang*

*Stanford University** *and Massachusetts Institute of Technology*†

This paper provides an overview of the resource sharing networks introduced by Massoulié and Roberts [20] to model the dynamic behavior of Internet flows. Striving to separate the model class from the applications that motivated its development, we assume no prior knowledge of communication networks. The paper also presents an open problem, along with simulation results, a formal analysis, and a selective literature review that provide context and motivation. The open problem is to devise a policy for dynamic resource allocation that achieves what we call hierarchical greedy ideal (HGI) performance in the heavy traffic limit. The existence of such a policy is suggested by formal analysis of an approximating Brownian control problem, assuming that there is "local traffic" on each processing resource.

1. Introduction. We consider the resource sharing networks, also called bandwidth sharing models, connection-level models, or flow-level models, that were introduced by Massoulié and Roberts [20] to study the dynamic behavior of Internet flows. This elegant model class may also prove useful in other application domains, and is of mathematical interest in its own right, so we shall avoid terminology that is specific to communication networks. In particular, we refer to the entities being processed in the model as *jobs*, rather than files or documents or flows, and speak of processing *resources* rather than links or servers. Verloop et al. [23] introduced the context-neutral term "resource sharing networks" to describe the Massoulié-Roberts model class, and we shall follow that usage.

Following [14], we assume that job sizes are exponentially distributed except where more general distributions are explicitly mentioned, and we assume that there is "local traffic" on each of the network's processing resources. We imagine a system manager who dynamically allocates resource capacities to jobs whose processing is not yet complete. The performance

Received September 2013.

Keywords and phrases: Bandwidth sharing models, dynamic resource allocation, entrainment, heavy traffic analysis, hierarchical, greedy ideal performance, resource sharing networks.

measure on which we focus is the steady-state expected total job count, or equivalently, the steady-state average delay experienced by arriving jobs, irrespective of their type.

Following a standard recipe, we formulate a Brownian control problem (BCP) that formally approximates the system manager's dynamic allocation problem under heavy traffic conditions. Given our local traffic assumption, there exists a control policy for the approximating BCP that has the following two properties: first, the backlog of work for each resource is minimal at each point in time, which is equivalent to saying that no resource's capacity is ever underutilized while there is work for that resource in the system; and second, the total number of jobs waiting at each point in time is the minimum number consistent with the vector of workloads for the various resources. This is called the hierarchical greedy (HG) policy for the approximating BCP, and its associated performance measure is referred to as *HG performance*. In the context of our original resource sharing network, we define an analogous performance target and call it *hierarchical greedy ideal* (*HGI*) performance. Loosely phrased, the open problem referred to above is the following: to formulate a control policy for the resource sharing network that achieves HGI performance in the heavy traffic limit.

We study three small but representative examples in which all resources have identical load factors, treating the common load factor as a variable parameter. For each example, HGI performance is estimated via simulation, and compared against the performance of proportionally fair (PF) resource allocation, which is the most commonly cited and most thoroughly studied dynamic allocation scheme for resource sharing networks. For load factors ranging from 0.80 to 0.95, HGI performance represents a 20–40 percent improvement over PF performance in our examples. A dynamic allocation scheme called UFOS (mnemonic for *utilization first, output second*) approximates HGI performance quite well in our three examples, but a fourth example due to R. Srikant shows that UFOS is not viable in general as a means of approaching the hierarchical greedy ideal.

The remainder of the paper is organized as follows. Sections 2 through 4 identify the model class under study, introduce our three examples, and establish notation and terminology. Sections 5 and 6 are devoted to proportional fairness and its relationship to what we call baseline performance, and Section 7 develops some basic theory related to workload. Ideas related to HGI performance are developed in Sections 8 through 11, including a recapitulation of the approximating Brownian control problem in Section 9. Section 12 explains the UFOS algorithm by which HGI performance is approximated in our small examples. Sections 13 and 14 discuss the inadequacy

of UFOS as a general method, and summarize our current state of knowledge. The paper concludes with two short technical appendices.

2. Resource sharing networks. In the general model to be considered, jobs of types $1, \ldots, J$ arrive according to independent Poisson processes at rates $\lambda_1, \ldots, \lambda_J$, and for concreteness we assume that the system under study is initially empty. Each type j arrival has a *size* that is drawn from a type-specific distribution with mean $m_j > 0$ $(j = 1, \ldots, J)$, and the job sizes for the different types form J mutually independent sequences of independent and identically distributed random variables. In the usual way, let $\mu_j = m_j^{-1}$ $(j = 1, \ldots, J)$. Unless something is explicitly said to the contrary, we assume the job size distribution to be exponential for each type.

The processing system is composed of resources numbered $1, \ldots, I$, with associated capacities $C_1, \ldots, C_I$ (The significance of resource capacities will be explained shortly.) The processing of a job is accomplished by allocating a *flow rate* to it over time: a job departs from the system when the integral of its allocated flow rate equals its size. In general, the processing of a job consumes the capacity of several different resources simultaneously, as follows. There is given a non-negative $I \times J$ matrix $A = (A_{ij})$, and each unit of flow allocated to type j jobs consumes the capacity of resources $1, \ldots, I$ at rates $A_{1j}, \ldots, A_{Ij}$, respectively. Thus, denoting by $x = (x_1, \ldots, x_J)$ the vector of total flow rates allocated to the various job types at a given time, Ax is the corresponding vector of capacity consumption rates for the various resources, and x must satisfy the capacity constraint $Ax \leq C = (C_1, \ldots, C_I)$. We use the terms "capacity allocation" and "flow rate allocation" interchangeably. The former term is standard in the literature, where the system manager's task is usually described as one of *dynamic capacity allocation*.

In the canonical application to Internet modeling, the job types correspond to files that require transmission over different routes, the resources correspond to transmission links, the job sizes are interpreted as file sizes, and the capacity consumption rate A_{ij} is either 1 or 0, depending on whether or not link i is part of the route used by jobs of type j. Positive values of A_{ij} other than 1 may also be meaningful, specifically in the reduced representation of systems with multi-path routing; see Section 5.5 of [14].

As in [14], we assume the following throughout this paper: for each $i \in \{1, \ldots, I\}$ there is a $j \in \{1, \ldots, J\}$ such that $A_{ij} > 0$ and $A_{kj} = 0$ for $k \neq i$. That is, for each resource i there is a type j whose processing consumes the capacity of resource i but *not that of any other resource*. In the context of communication networks, this is referred to as a "local traffic" assumption, for obvious reasons. Readers will see that the local traffic assumption plays a crucial role in our analysis.

We denote by $n_j(t)$ the number of type j jobs residing in the system at time t, calling $n(t) = (n_1(t), \ldots, n_J(t))$ the *job count vector* or *state vector* and calling $\{n(t), t \geq 0\}$ the *job count process*. A *control* takes the form of a flow rate vector $x = (x_1, \ldots, x_J)$ for each time $t \geq 0$. Of course, controls must be suitably non-anticipating. For our purposes it will suffice to consider only stationary Markov control policies, meaning that the flow rate vector employed at each time t is a deterministic function of $n(t)$. That is, we define an *admissible control* as a function $x(n) = (x_1(n), \ldots, x_J(n))$ such that $x(n) \in \Phi(n)$ for each state n, where

$$\Phi(n) = \{x \geq 0 : Ax \leq C \text{ and } x_j = 0 \text{ if } n_j = 0\}. \tag{2.1}$$

A number of other terms, including "dynamic allocation scheme" and "resource sharing policy," will be used as synonyms for "control" later in the paper.

In the definition of a control that we have advanced, $x_j(n)$ is interpreted as the total flow rate allocated to type j jobs when the system is in state n, without regard to how that total flow is divided among the type j jobs residing in the system. That finer information is irrelevant for our purposes, because the performance measures on which we focus are expressed solely in terms of job counts, and we assume exponential (memoryless) job size distributions, so the division of the total flow rate among individual jobs of a given type does not affect the probabilistic evolution of the job count process $\{n(t),\ t \geq 0\}$. Also, because the flow rate vector chosen at each time t is only allowed to depend on the state vector $n(t)$, we implicitly assume that the system manager either cannot observe job sizes or else is not allowed to base capacity allocations on that information.

At some points in this paper mention will be made of resource sharing networks with general (non-exponential) job size distributions, and in such models it *is* necessary to specify how the flow rate allocated to a job type is divided among individual jobs of that type. In all such cases we assume an *equal sharing* rule, which means that the type j flow rate is divided equally among all type j jobs residing in the system. This assumption is standard in the literature. (With one resource and one job type, it corresponds to the usual processor sharing discipline.)

For future purposes let

$$M = \operatorname{diag}(m_1, \ldots, m_J) \text{ and } \rho_i = \frac{1}{C_i} \sum_{j=1}^{J} A_{ij} \lambda_j m_j \text{ for } i = 1, \ldots, I. \tag{2.2}$$

The sum in the definition of ρ_i is the expected amount of resource i capacity needed to process jobs of all types that arrive during one time unit, and we

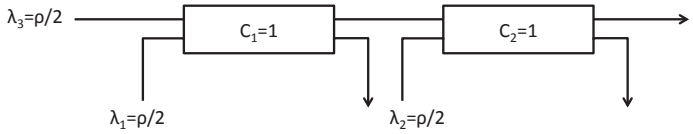

FIG 1. *Two-Link Linear Network (2LLN).*

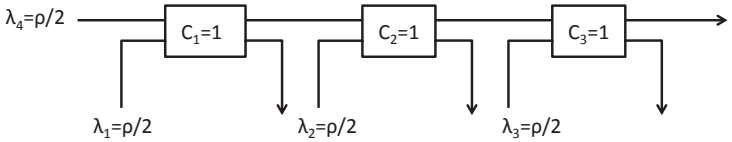

FIG 2. *Three-Link Linear Network (3LLN).*

divide that by the amount of resource i capacity that is available per time unit. Thus we call ρ_i the *load factor* for resource i. (Such ratios are also called *traffic intensity parameters* in queuing theory.)

In the basic network model that we have described here, the collection of resources required to process any given job type is fixed. In the language of communication networks, it is a model with *single-path routing*. In contrast, one may consider a more general model in which several different processing modes exist for a given job type, with each mode involving a different combination of resources. In communication networks, the term *multi-path routing* is used to describe that more general setup. Remarkably, any resource sharing model with multi-path routing is *exactly* equivalent, not just approximately equivalent or asymptotically equivalent, to another model with only single path routing; that equivalence is explained in Section 5.5 of [14].

3. Three examples. Figures 1 through 3 portray three examples of resource sharing networks that will be discussed later in this paper, each of which satisfies the local traffic assumption enunciated in Section 2. The first two examples are what Massoulié and Roberts [20] call *linear networks*, involving local traffic on each of the I resources plus another job type that uses all of the resources; to state the obvious, $I = 2$ and $I = 3$ in Figures 1 and 2, respectively. Our third example (Figure 3) is a more complicated version of the second one, involving two additional job types that each require two of the three resources for their processing. Using Internet modeling language, we call the first two examples the *two-link linear network* (2LLN) and *three-link linear network* (3LLN), respectively; in parallel fashion, the final example is called the *complex three-link network* (C3LN).

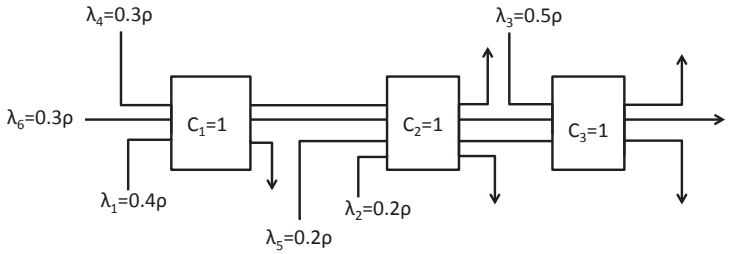

FIG 3. *Complex Three-Link Network (C3LN).*

In Figures 1 through 3 the Poisson arrival rates for the various job types are expressed as multiples of a variable parameter ρ, and as noted in the figures, we take $C_i = 1$ for each resource i as well. Finally, in each example *we assume that the job size distribution is exponential with mean 1 for each job type*. Defining A as the obvious matrix of zeros and ones for each of the examples (A is 2×3 in the first example, 3×4 in the second one, and 3×6 in the third), readers can easily verify that

$$\rho_i = \rho \text{ for all } i = 1, \ldots, I \text{ in all of our three examples.} \tag{3.1}$$

4. Stability and system performance. Because we restrict attention to stationary Markov controls (see Section 2), the job count process $\{n(t),\ t \geq 0\}$ evolves under any admissible control as a continuous-time Markov chain with stationary transition probabilities: transitions that increment n_j by one occur at rate $\lambda_j (j = 1, \ldots, J)$, transitions that decrement n_j by one occur at rate $\mu_j x_j(n)$ $(j = 1, \ldots, J)$, and all other transition intensities are zero. An admissible control will be called stable if the associated Markov chain is positive recurrent. It is known that there exist stable controls if and only if

$$\rho_i < 1 \text{ for all } i = 1, \ldots, I, \tag{4.1}$$

and (4.1) will be referred to hereafter as *the usual traffic condition*. For a proof that (4.1) is necessary for existence of a stable control, see page 1060 of [18]; with regard to sufficiency, we shall describe in Section 7 a specific control policy (proportional fairness) that is known to be stable when (4.1) holds. *Attention will be restricted hereafter to resource sharing networks that satisfy the usual traffic condition* (4.1). Throughout most of this paper, we use the term *heavy traffic* to mean that ρ_i is close to 1 for *every* resource i, but at the very end (Section 14), attention is directed to the broader scenario where ρ_i is close to 1 for *some*, but not necessarily all, resources i.

Given a stable admissible control, let $n(\infty)$ be a random variable whose distribution is the stationary distribution of the associated job count process. The performance measure on which we focus is

$$E\,(Tot) = E\left[\sum_{j=1}^{J} n_j(\infty)\right]. \tag{4.2}$$

In words, $E\,(Tot)$ is the steady-state *expected total* job count (under the specified control). By Little's law, the steady-state average delay for arriving jobs (that is, the average elapsed time in steady-state, between arrival of a job and completion of its processing) equals $E\,(Tot)$ divided by the total arrival rate $\lambda_1 + \cdots + \lambda_J$. Thus minimizing the steady-state average delay is equivalent to minimizing $E\,(Tot)$, and a given percentage reduction in $E\,(Tot)$, by one control versus another, ensures the same percentage reduction in steady-state average delay.

5. Proportional fairness. Given a job count vector n, the vector x of *proportionally fair* (PF) flow rate allocations solves the following problem:

$$\text{choose } x \in \Phi(n) \text{ to maximize } \sum_j n_j \log x_j, \tag{5.1}$$

where $\Phi(n)$ is defined by (2.1). PF allocations are non-extremal: every job type that is currently represented in the backlog of work gets *some* flow rate allocation, regardless of system status. The idea of proportional fairness originated in work by Kelly [16] and was raised to prominence by the influential paper [17]. A more general concept of *α-fair allocations* was advanced and analyzed by Mo and Walrand [21], but discussion will be restricted here to the original notion of proportional fairness.

Explaining the rationale for proportional fairness in digital communications is rather complicated, and we shall not attempt to do so. But a reader may reasonably ask in what sense does (5.1) lead to "fair" flow rate allocations, and there is no good answer to that question in the setting we consider, as indicated by the following quotations from [20]: "In this paper we argue that fairness should be of secondary concern and that the network should be designed rather to fulfill minimal quality of service requirements ... [Q]uality of service is manifested essentially by the time it takes to complete the document transfer. (p. 186)... User perceived quality of service may be measured by the response time of a given document transfer... The fact that this [transfer was achieved] 'fairly' is largely irrelevant and, moreover, totally unverifiable by the user (pp. 189–190)."

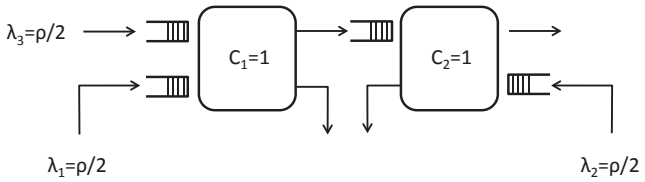

FIG 4. *Store-and-forward analog of the 2LLN.*

We shall take the view that "proportional fairness" is simply a name attached to a particular method for dynamic resource allocation. To evaluate this method, one must analyze the delay performance of PF and make comparisons against the delay performance of other allocation schemes. Assuming exponential job size distributions (as we do), [8] showed that PF allocations are stable whenever (4.1) holds. Of course, a stable control policy can still have undesirable delay characteristics, but results reviewed in the next section justify calling delay performance under PF at least "good."

In the context of communication networks, an important virtue of proportional fairness is that it can be implemented, at least approximately, by means of a distributed algorithm, requiring only local information for purposes of local decision making; see [17] for elaboration.

6. Baseline performance. The salient feature of resource sharing networks, relative to conventional queueing networks, is simultaneous resource possession. Conceptually, arriving jobs of any given type are stored in a type-specific buffer upon arrival, and are processed by means of a capacity "pipeline" (generally involving several resources) that completes the processing in a single hop.

If we think of the network resources as communication links that are combined to transmit jobs over different routes, then a potential alternative arrangement, commonly called *store-and-forward processing*, is to transmit the entire job over the first link on its route, store the job in an intermediate buffer, then transmit the job over the second link on its route, store it in another intermediate buffer, and so on to completion. This is illustrated for our two-link linear network in Figure 4, assuming that resource 1 is traversed first in the routing of type 3 jobs.

In Figure 4 there are three "external" buffers where arriving jobs of different types are stored, plus one "internal" buffer where partially completed jobs of type 3 are stored. For our complex three-link network (C3LN), originally pictured in Figure 3, the analogous store-and-forward representation would include six "external" buffers for newly arriving jobs of different types,

plus a total of four "internal buffers" for storing jobs of types 4 through 6 at intermediate stages of their routes. In our specification of a resource sharing network, no ordering is given for the resources involved in the processing of a given job type, so one could specify the "route" of any given type in several different ways, each of which gives rise to a different store-and-forward analog. However, the result cited below (Proposition 6.1) is invariant to the ordering.

Each of the buffers in a store-and-forward network is *associated* with a specific resource, namely, the resource from which the jobs stored in that buffer need processing next. Before we can analyze the delay performance of a store-and-forward network, a method must be specified for dividing the capacity of each resource among the jobs waiting in the resource's associated buffers. In communications modeling, the method widely viewed as "standard" is *processor sharing* (PS), in which the capacity of each resource is shared equally among all the individual jobs residing in the resource's associated buffers. Under PS, then, the capacity of a resource is divided among its associated buffers in proportion to the job counts in those buffers; to repeat an earlier point, it is irrelevant for our purposes how the capacity allocated to a given buffer is divided among the individual jobs in that buffer, because (a) the performance measures on which we focus are expressed solely in terms of job counts, and (b) we assume exponential job size distributions, so the division of the total flow rate among individual jobs of a given type does not affect the probabilistic evolution of the job count process.

With store-and-forward processing, our system has the structure of a multi-class queueing network, with each resource functioning as a single-server station, and with one "customer class" defined for each buffer (that is, one customer class defined for each job type at each stage on its route), but with the following non-standard feature: the size of a job, which is drawn from a type-specific exponential distribution upon arrival to the network, remains the same at each stage of its processing, so the "service times" of an individual job at successive processing stages are perfectly correlated. To get a queueing network model of standard type, we shall treat the service times of a single job at successive processing stages as *independent* random variables, each of them having the appropriate type-specific exponential distribution. This is a version of what is called *Kleinrock's independence assumption*, or *Kleinrock's independence approximation*, in the literature of communication networks; see, for example, page 115 of [19]. Section 4.1 of [15] contains a persuasive but non-rigorous argument that the independence approximation does not actually affect the product-form result cited below

(that is, that the network's steady-state distribution is the same whether a job's successive service times are perfectly correlated or are independent).

Hereafter, when we refer to the *SFPS analog of a resource sharing network*, this means the store-and-forward system described at the beginning of this section, with processor sharing used to dynamically allocate the capacity of each resource, and with the added independence approximation explained in the previous paragraph. Such a multi-class queueing network is known to have a product form stationary distribution, because each of its single-server stations is what Kelly [15] called a "symmetric queue." To be specific, it follows from Theorems 3.7 and 3.8 of [15] that the steady-state distribution of the SFPS analog can be specified via the constructive procedure immediately below; the same result can be found in [1], Chapter 2 of [6], and Chapter 4 of [7].

For the construction, let us denote by $N_1, \ldots, N_I$ the numbers of individuals that belong to "populations" numbered $1, \ldots, I$ and suppose that each individual is given a "label" that belongs to the set $\{1, \ldots, J\}$. (These "individuals" correspond to jobs being served in our multi-class queueing network. One interprets N_i as the total number of jobs occupying station i in steady state, and the label given to an individual is interpreted as the type of the corresponding job.) The probabilistic assumptions are as follows:

(a) the population sizes $N_1, \ldots, N_I$ are independent random variables;
(b) N_i is geometrically distributed with mean $\rho_i/(1-\rho_i)$ for $i = 1, \ldots, I$; and
(c) the probability that an individual belonging to population i is given label j equals $A_{ij}\lambda_j m_j / \sum_{k=1}^{J} A_i \lambda_k m_k (i = 1, \ldots, I$ and $j = 1, \ldots, J)$, independent of how other individuals are labeled.

Denoting by L_j the total number of individuals that are given label j, we see that L_j is the sum of I independent geometrically distributed random variables $(j = 1, \ldots, J)$, and that $L_1, \ldots, L_J$ are generally *not* independent. For the store-and-forward analog of a resource sharing network, let us denote by $n(t) = (n_1(t), \ldots, n_J(t))$ the job count vector at time t, each component of which involves a sum over buffers containing a given job type, and let $n(\infty)$ be a random vector having the associated stationary distribution. The first statement of the following proposition articulates the "product form" result referred to above, and the second statement follows from the fact that $L_1 + \cdots + L_J = N_1 + \cdots + N_I$.

PROPOSITION 6.1. *For the SFPS analog of a resource sharing network, $n(\infty)$ is distributed as the vector $L = (L_1, \ldots, L_J)$ constructed above, and*

hence

$$E\left(Tot\right) = \sum_{i=1}^{I} \left(\frac{\rho_i}{1-\rho_i} \right). \tag{6.1}$$

As noted earlier, processor sharing is widely viewed as a fair and reasonable mechanism for allocating resource capacities in store-and-forward networks, and there is no *a priori* reason to think that resource sharing (that is, simultaneous resource possession mandated by an absence of internal buffering) is either more or less favorable than a store-and-forward protocol with regard to delay performance, so we shall refer to (6.1) hereafter as *baseline performance*. Recent results by Kang et al. [14] and Shah et al. [22] show that, in heavy traffic, the delay performance of a resource sharing network with flow rate allocation via proportional fairness (see Section 5) is approximately baseline performance; the next paragraph explains their results in more detail.

For a resource sharing network of the kind considered here, with proportionally fair resource allocation, [14] proves that a properly scaled version of the job count process converges weakly in heavy traffic to a particular diffusion limit. That limit is a semi-martingale reflected Brownian motion (SRBM) with what are called skew-symmetric data, and so its stationary distribution has a product form. More specifically, the stationary distribution of the SRBM is the heavy traffic limit of the product form stationary distribution for the original network's SFPS analog (see Proposition 6.1). Kang et al. [14] do not deal directly with convergence of stationary distributions, but [22] shows that their heavy traffic limit can be interchanged with the $t \to \infty$ limit. That is, if one considers a sequence of networks in heavy traffic, assuming that each of them satisfies the usual traffic condition (4.1), their normalized stationary distributions converge to the stationary distribution of the SRBM that is their heavy traffic process limit.

Table 1 presents simulation results which are consistent with that theoretical analysis, focusing exclusively on the bottom-line performance measure $E(Tot)$. For each of the three examples introduced in Section 3, figures in the column labeled "PF simulation" were derived using Monte Carlo simulation and proportionally fair resource allocation, and figures in the "baseline" column were computed using formula (6.1).

Table 1 reports only the aggregate performance measure $E(Tot)$ for our three examples, but the theoretical results cited above establish the following stronger conclusion: in heavy traffic, *the entire stationary distribution* of job counts for a resource sharing network with PF allocations is well approximated by the stationary job count distribution of the network's SFPS

TABLE 1
$E(Tot)$ comparison: Baseline versus proportional fairness for our three examples

Example	Load Factor ρ	$E(Tot)$ Values Baseline	$E(Tot)$ Values PF Simulation	$\frac{\text{Baseline} - \text{PF}}{\text{Baseline}}$
	0.80	8.00	7.33	8%
2LLN	0.90	18.00	17.17	5%
	0.95	38.00	37.11	2%
	0.80	12.00	10.56	12%
3LLN	0.90	27.00	25.2	7%
	0.95	57.00	53.97	5%
	0.80	12.00	10.34	14%
C3LN	0.90	27.00	24.75	8%
	0.95	57.00	54.29	5%

TABLE 2
Detailed comparison: Baseline versus proportional fairness for the 3LLN

Comparisons for $\rho = 0.80$									
$E(n1)$	$E(n2)$	$E(n3)$	$E(n4)$	$\sigma(n1)$	$\sigma(n2)$	$\sigma(n3)$	$\sigma(n4)$	$E(Tot)$	
2.00	2.00	2.00	6.00	2.45	2.45	2.45	4.24	12.00	Baseline
2.00	1.97	1.98	4.62	2.46	2.40	2.44	3.95	10.56	PF Simulation
Comparisons for $\rho = 0.90$									
$E(n1)$	$E(n2)$	$E(n3)$	$E(n4)$	$\sigma(n1)$	$\sigma(n2)$	$\sigma(n3)$	$\sigma(n4)$	$E(Tot)$	
4.50	4.50	4.50	13.50	4.97	4.97	4.97	8.62	27.00	Baseline
4.52	4.45	4.45	11.77	5.02	4.93	4.94	8.37	25.20	PF Simulation
Comparisons for $\rho = 0.95$									
$E(n1)$	$E(n2)$	$E(n3)$	$E(n4)$	$\sigma(n1)$	$\sigma(n2)$	$\sigma(n3)$	$\sigma(n4)$	$E(Tot)$	
9.50	9.50	9.50	28.50	9.99	9.99	9.99	17.30	57.00	Baseline
9.37	9.27	9.28	26.05	9.88	9.66	9.66	16.53	53.97	PF Simulation

analog. Table 2 presents more detailed simulation results for the 3LLN example (see Figure 2) which give credence to that view. In that table, simulation estimates are provided for both the mean and the standard deviation of the stationary job count distribution for each of the network's four job types, along with baseline values for each of those quantities; in the obvious way, the baseline values are computed from the product form stationary distribution of the SFPS analog (see Proposition 6.1). The simulation estimates agree quite well generally with the baseline values, and the agreement tends to improve as the system load factor increases.

Kang et al. [14] showed that their heavy traffic diffusion limit under proportional fairness, which has a product form stationary distribution, remains the same when job size distributions are mixtures of exponentials. Based on that extension, one may plausibly conjecture that the product form approx-

imation under proportional fairness, corresponding to what we have called baseline performance, remains valid with general (non-exponential) job size distributions.

Thus far we have spoken of "baseline performance" only as an approximation, or benchmark, but the analysis of Zachary [26] shows that baseline performance is *exactly achievable* in any resource sharing network that satisfies the usual traffic condition (4). Zachary's analysis, which generalizes earlier work by Bonald and Proutiére [4, 5], shows that there exists a dynamic allocation scheme (see below) under which the stationary job count distribution of the resource sharing network coincides *exactly* with the product form distribution of the network's SFPS analog, and this remains true even with general (non-exponential) job size distributions. Thus, even if the system manager's objective is to minimize $E\left[h\left(n(\infty)\right)\right]$ for some arbitrary cost function $h(\cdot)$, we know that *the baseline value of that performance measure* (that is, its value under the product form distribution defined by construction earlier in this section) *is an upper bound on the minimum achievable value.*

The dynamic allocation scheme referred to in the previous paragraph is determined as follows. In the final paragraph of his paper, Zachary [26] observes that a resource sharing network of the kind considered here is an example of his multi-class network model "with no internal transitions," and hence the partial balance equations appearing in his Theorem 2 reduce to the detailed balance equations numbered (16) in his paper. By specifying the equilibrium distribution $\pi(\cdot)$ in those equations to be the product-form distribution described earlier in this section, and specifying the arrival rates for jobs of types $1, \ldots, n$ to be the constants $\lambda_1, \ldots \lambda_n$, irrespective of the current state, one can simply solve for the departure rates of the various job types in various systems states, and those departure rates immediately determine the state-dependent flow rate allocations (or capacity allocations) for the various job types. Of course, it must be verified that those allocations satisfy the capacity constraints for all resources, and doing so is a straightforward task.

7. Nominal and actual workload processes. The main question that we wish to address in the remainder of this paper is the following: Can one improve significantly on baseline performance, assuming that $E(Tot)$ is the performance measure of interest? To address that question, some further basic theory is needed. We first define the *nominal workload* for resource i at time t as follows:

$$\hat{w}_i\left(t\right) = \sum_{j=1}^{J} A_{ij} n_j(t) m_j \text{ for } i = 1, \ldots, I. \tag{7.1}$$

In contrast, we denote by $w_i(t)$ the *actual workload* for resource i at time t, which means the total amount of capacity required from resource i to complete the processing of jobs residing in the system at time t. (It is perhaps worth noting that in [14] the name "workload" is used for what we call nominal workload.)

Seeking to express the verbal definition of actual workload in terms of model primitives (that is, in terms of arrival processes, job size random variables, resource consumption rates and resource capacities), we proceed as follows. First, for each resource i and each $t \geq 0$, let $\ell_i(t)$ denote the total amount of resource i capacity required to process all jobs that arrive over $[0,t]$. (The letter ℓ is mnemonic for *load*.) This quantity can be expressed in terms of arrival processes and job size random variables for the various job types j, plus the resource consumption rates A_{ij}. Then one has

$$w_i(t) = \ell_i(t) - \sum_{j=1}^{J} A_{ij} \int_0^t x_j(s)ds = \xi_i(t) + u_i(t), \ t \geq 0, \tag{7.2}$$

where

$$\xi_i(t) = \ell_i(t) - C_i t \text{ and } u_i(t) = \int_0^t \left[C_i - \sum_{j=1}^{J} A_{ij} x_j(s) \right] ds, \ t \geq 0. \tag{7.3}$$

We call $\{\xi(t), t \geq 0\}$ the *netflow* process for resource i, and interpret $u_i(t)$ as the total amount of resource i capacity that goes *unused* over $[0,t]$. Note that $\xi_i(t)$ is a random variable defined directly in terms of model primitives, without reference to the control chosen by the system manager, whereas $u_i(t)$ is dependent on the chosen control. Because $\{w_i(t), t \geq 0\}$ is by definition a non-negative process, one has $u_i(t) \geq -\min\{\xi_i(s), 0 \leq s \leq t\}$, which implies

$$w_i(t) \geq w_i^*(t), \tag{7.4}$$

where

$$w_i^*(t) = \xi_i(t) - \min\{\xi_i(s), 0 \leq s \leq t\}, \quad t \geq 0. \tag{7.5}$$

We call $\{w_i^*(t), t \geq 0\}$ the *minimum workload process* for resource i, observing that (7.4) holds with *equality* if and only if, over the entire time interval [0,t], resource i is able to work at full capacity whenever there is work for it to do in the system.

The process $w^*(t) = (w_1^*(t), \ldots, w_I^*(t))$ is defined directly in terms of model primitives, and does not depend on the control chosen by the system manager. Using a standard time reversal argument, as in Section 1.10 of [12], we have that $w^*(t) \sim M(t)$ for each fixed $t > 0$, where "$\sim$" denotes equality

in distribution, $M(t) = (M_1(t), \ldots, M_I(t))$, and

$$M_i(t) = \max\{\xi_i(s), 0 \le s \le t\} \text{ for } t \ge 0 \text{ and } i = 1, \ldots, I. \tag{7.6}$$

(The time-reversal argument requires only that the process $\xi(t) = (\xi_1(t), \ldots, \xi_I(t))$, $t \ge 0$, have stationary independent increments and $\xi(0) = 0$.) It follows that the random vectors $w^*(t)$ increase stochastically to a finite limit $w^*(\infty)$ as $t \uparrow \infty$. Also, under any stable admissible control, the workload vector $w(t) = (w_1(t), \ldots, w_I(t))$ converges in distribution as $t \uparrow \infty$ to a finite limit (see Appendix A) that we shall denote $w(\infty)$, and by (7.4) we can define $w(\infty)$ and $w^*(\infty)$ on a common probability space in such a way that

$$w(\infty) \ge w^*(\infty). \tag{7.7}$$

Given a stable admissible control, let $w(t)$ be the associated I-dimensional actual workload process, and let $\hat{w}(t)$ be the associated I-dimensional nominal workload process. We can rewrite (7.1) in vector-matrix form as

$$\hat{w}(t) = AMn(t), \ t \ge 0. \tag{7.8}$$

Also, from the memoryless property of the exponential distribution, we have that $\hat{w}_i(t) = E[w_i(t) | n(t)]$ for all i and t, which can be expressed in vector form as

$$\hat{w}(t) = E[w(t) | n(t)], \ t \ge 0. \tag{7.9}$$

8. Minimum possible cost rate given nominal workload. Hereafter we denote by e the J-vector of ones, and by $\mathbf{R}_+^k$ the non-negative orthant of k-dimensional Euclidean space. Formula (8.1) below defines a function $f\colon \mathbf{R}_+^I \to \mathbf{R}_+^J$ for which an interpretation will be provided shortly.

PROPOSITION 8.1. *For each $w \in \mathbf{R}_+^I$ the set $\{z \in \mathbf{R}_+^J : AMz = w\}$ is non-empty, and so it is meaningful to define*

$$f(w) = \min\{e \cdot z : AMz = w, z \in \mathbf{R}_+^J\} \text{ for } w \in \mathbf{R}_+^I. \tag{8.1}$$

Moreover, there exists a continuous function $g : \mathbf{R}_+^I \to \mathbf{R}_+^J$ such that

$$g(w) = \operatorname{argmin}\{e \cdot z : AMz = w, z \in \mathbf{R}_+^J\} \text{ for } w \in \mathbf{R}_+^I. \tag{8.2}$$

PROOF. Restated in different terms, our local traffic assumption says that there exist I columns of the capacity consumption matrix A that constitute an $I \times I$ diagonal sub-matrix with strictly positive diagonal elements. The

same is then true for AM, from which the first statement of the proposition is immediate. The second statement of the proposition says that the optimal solution of a certain linear program can be chosen as a continuous function of the right-hand side values. It follows from the Basis Decomposition Theorem in Section 3 of [25], as those authors explain immediately before the theorem statement. □

For an interpretation of $f(w)$, it is useful to think of $e \cdot n$ as the *cost rate* associated with a job count vector n, so our performance measure $E(Tot)$ represents the steady-state average cost rate under a given policy. Because $n(t) \geq 0$ for any $t \geq 0$, the definitions (7.8) and (8.1) imply

$$e \cdot n(t) \geq f(\hat{w}(t)) \text{ for all } t \geq 0. \tag{8.3}$$

That is, no job count vector which yields a given nominal workload vector w can achieve a cost rate smaller than $f(w)$. Moreover, if one ignores the distinction between integer and non-integer job counts (which is a minor distinction in heavy traffic), then the function value $f(w)$ in (8.1) can be described as the *minimum possible cost rate* given that the nominal workload vector is w. (This interpretation ignores the fact that not all $w \in \mathbf{R}_+^I$ can occur as nominal workload vectors.) Perhaps surprisingly, $f(\cdot)$ is not necessarily monotone in all of its arguments; see below for elaboration.

It is instructive to consider the form of the function $f(\cdot)$ for the first two examples specified in Section 3. For our two-link linear network (Figure 1) one has

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

and M is the 3×3 identity matrix, from which one obtains the following: for any workload vector $w = (w_1, w_2)$, the minimizing choice of z in (8.1) is $z = \left((w_1 - w_2)^+, (w_2 - w_1)^+, w_1 \wedge w_2\right)$, which gives $f(w) = w_1 \vee w_2$. That is, to minimize the cost rate $z_1 + z_2 + z_3$ given w, one takes z_3 (the number of waiting jobs that require both resources for their processing) as large as possible. In this case we see that $f(\cdot)$ is monotone increasing in both arguments. For our three-link linear network (Figure 2), one has

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

and M is the 4×4 identity matrix, from which one obtains

$$f(w) = \bigwedge_{i=1}^{3} w_i + \sum_{i=1}^{3} (w_i - \bigwedge_{k=1}^{3} w_k).$$

In this case $f(\cdot)$ is *not* monotone, as shown by the following: for $w = (2, 2, 1)$ the minimizing choice of z in (8.1) is (1,1,0,1), and for $w = (2, 2, 2)$ it is (0,0,0,2), implying that $f(2, 2, 1) = 3$ but $f(2, 2, 2) = 2$. That is, a lower cost rate (lower total job count) can be achieved with a higher workload for resource 3, because then it becomes possible to hold all work in the form of jobs that need processing by all three resources.

9. Approximating Brownian system model. In Section 1 it was said that we follow a "standard recipe" in formulating a Brownian approximation of the system manager's dynamic allocation problem. To be more specific, we adopt the framework developed in Section 5 of Harrison [11], referred to hereafter as H2000. The general resource sharing network described in Section 2 of this paper is a straightforward example of what is called a "stochastic processing network" in H2000: in a resource sharing network we have I processing resources, J different "materials" being processed (namely, jobs of the J different types), and J "processing activities" (namely, the processing of the J different job types). At any given time t, it is the flow rate allocated to type j jobs that constitutes the "activity level" for activity j. In our context, the input-output matrix R identified in H2000 is the $J \times J$ diagonal matrix M^{-1}, because the stock of material j on hand (that is, the number of type j jobs residing in the system) is decreased at an average rate of $\mu_j x_j$ when activity j is conducted at level x_j. Finally, the capacity of resource i is consumed at rate $A_{ij}x_j$ when activity j is conducted at level x_j.

Conforming to the framework of H2000, we state our heavy traffic assumption in the following form: there exist *nominal arrival rates* $\lambda_1^*, \ldots, \lambda_J^*$ and a small parameter $\varepsilon > 0$ such that

$$\theta_j = (\lambda_j - \lambda_j^*)/\varepsilon \text{ is of moderate size (order 1) for each } j = 1, \ldots, J, \tag{9.1}$$

and moreover,

$$AM\lambda^* = C. \tag{9.2}$$

Conditions (9.1) and (9.2) can be expressed verbally as follows: the nominal arrival rates are close to the actual arrival rates, and taken in aggregate, they load each resource to exactly its capacity. (Given our local traffic assumption, the H2000 notion of "heavy traffic" holds if and only if all of the actual load factors $\rho_1, \ldots, \rho_J$ are close to 1.) The *nominal activity levels* defined in H2000 are simply $x_j^* = \lambda_j^*/\mu_j$ for $j = 1, \ldots, J$ and then we re-express the system manager's chosen control in the following form, using

the simplified notation $x(t) = (x_1(t), \ldots, x_J(t))$ to denote the vector of flow rate allocations chosen at time t:

$$y_j(t) = x_j^* t - \int_0^t x_j(s)ds \text{ for } j = 1, \ldots, J \text{ and } t \geq 0. \tag{9.3}$$

Elements of the vector $y(t)$ re-express the system manager's allocations to the various job types as *cumulative decrements* from the nominal allocations. Now the small parameter ε is used as a scaling constant in the following definitions:

$$Z(t) = \varepsilon n(\varepsilon^{-2}t), \quad Y(t) = \varepsilon y(\varepsilon^{-2}t), \text{ and } U(t) = \varepsilon u(\varepsilon^{-2}t) \tag{9.4}$$

for $t \geq 0$. That is, we define Z, Y and U as diffusion-scaled versions of the job count process n, the chosen control y, and the unused capacity process u, respectively.

It follows from (9.3), (9.4), the definition of $u(\cdot)$ in (7.3), and the defining characteristics of the nominal arrival rates λ_j^* that

$$U(t) = AY(t) \text{ for } t \geq 0. \tag{9.5}$$

The key relationship for the approximate system model developed in Section 5 of H2000 is

$$Z(t) = X(t) + RY(t) = X(t) + M^{-1}Y(t), \ t \geq 0, \tag{9.6}$$

where X is a J-dimensional Brownian motion having drift vector $\theta = (\theta_1, \ldots, \theta_J)$ and a particular covariance matrix Σ that need not concern us here. We also have the obvious requirements that

$$U(\cdot) \text{ is non-decreasing with } U(0) = 0, \tag{9.7}$$

and

$$Z(t) \geq 0 \text{ for all } t \geq 0. \tag{9.8}$$

In the approximating Brownian system model, the Brownian motion X is taken as primitive, and the system manager must choose a control Y that is non-anticipating with respect X, subject to the constraints (9.7) and (9.8), where $U(\cdot)$ and $Z(\cdot)$ are defined by (9.5) and (9.6), respectively. Up to now nothing has been said about the system manager's objective, but let us suppose it is to

$$\text{minimize } E[e \cdot Z(\infty)]. \tag{9.9}$$

That is, we restrict attention to control policies under which Z has a steady-state distribution, denoting by $Z(\infty)$ a random variable which has that distribution, and seek to minimize the analog of $E(Tot)$ for the Brownian model. The addition of this objective to the system equations (9.5)–(9.8) gives us a *Brownian control problem* (BCP) that approximates the dynamic control problem discussed earlier for the original resource sharing network.

10. Hierarchical greedy control in the Brownian model. The Brownian approximation of a stochastic processing network is invariably simpler than the original, "exact" model that it replaces, and one manifestation of that simplicity is the existence of an *equivalent workload formulation* of the approximating BCP, the general theory of which was developed by Harrison and Van Mieghem [13]. Rather than recapitulate all of that theory, a few simple observations will suffice for our purposes here. First, let us define

$$W(t) = AMZ(t), \ t \geq 0. \tag{10.1}$$

Recall that Z is interpreted as a diffusion-scaled version of the job count process in our resource sharing network. Thus, comparing (7.8) and (10.1), one is led to interpret W as a diffusion-scaled version of the nominal workload process $\hat{w}$ that was defined in Section 8. However, given our assumption of exponential job size distributions, the diffusion-scaled difference between nominal and actual workload processes vanishes in the heavy traffic limit; see Appendix A for a sketch of the standard argument supporting that conclusion. Expressing that state of affairs more loosely, one may say that the distinction between nominal and actual workloads is negligible in heavy traffic, so W will be called simply the *workload process* (without any modifier) for our approximating Brownian system model. Multiplying both sides of (9.6) by AM, then substituting (9.5) and (10.1), we have the key relationship

$$W(t) = B(t) + U(t), \ t \geq 0, \ \text{where } B(t) = AMX(t), \ t \geq 0. \tag{10.2}$$

Now (10.1) implies that $W(\cdot) \geq 0$, so for each $i = 1, \ldots, I$ and each $t \geq 0$, the smallest possible value for $U_i(t)$ is

$$U_i^*(t) = -\min_{0 \leq s \leq t} B_i(s), \tag{10.3}$$

which corresponds to the minimum workload process

$$W_i^*(t) = B(t) + U_i^*(t). \tag{10.4}$$

Defining $W^*(t) = (W_1^*(t), \ldots, W_I^*(t))$ in the obvious way, we can invoke Proposition 8.1 to define a continuous process

$$Z^*(t) = g(W^*(t)), \ t \geq 0, \tag{10.5}$$

which gives us $Z^*(\cdot) \geq 0$ and

$$e \cdot Z^*(t) = f(W^*(t)), \ t \geq 0. \tag{10.6}$$

Inverting the fundamental system equation (9.6), the corresponding control is

$$Y^*(t) = M\left[X(t) - Z^*(t)\right], \ t \geq 0. \tag{10.7}$$

Equations (10.3), (10.4), (10.5) and (10.6) together define an admissible control Y^* for the approximating BCP that has two distinguishing features. First, it minimizes cumulative unused capacity of all resources simultaneously at all points in time, thus achieving the minimum workload process W^*. The proof of Proposition 8.1 shows that our local traffic assumption is essential for existence of such a control: it ensures that *any* non-negative workload vector w satisfies $w = AMz$ for *some* non-negative job count vector z; expressing the same thing in different words, it ensures that the potential state space for the workload process W in the Brownian system model is the *entire* orthant $\mathbf{R}_+^I$.

The second distinguishing feature of the control Y^* is the following: at every time t it achieves the lowest cost rate $e \cdot Z(t)$ that is possible given the constraints imposed by maximum resource utilization. We call Y^* the *hierarchical greedy* (HG) control, or hierarchical greedy policy, for our approximating BCP, because it first focuses myopically (greedily) on maximizing resource utilization, and then, given the constraints imposed by that dominant concern, configures the backlog of work so that the associated cost rate is minimized. That is, the control Y^* represents or embodies a hierarchical strategy in which resource utilization is primary and job count is secondary. The associated steady-state performance measure $E\left[e \cdot Z^*(\infty)\right] = E\left[f(W^*(\infty))\right]$ will be referred to hereafter as *HG performance* for our approximating BCP.

It should be emphasized that HG performance is *not* necessarily optimal performance in the BCP, because the minimum-possible-cost-rate mapping $f(\cdot)$ is not necessarily monotone (see Section 8). That is, greedily maximizing resource utilization may actually be inconsistent with minimizing steady-state total job count, but one feels intuitively that reducing workload will tend to have a favorable effect on job count as well.

The maximum-utilization aspect of the HG control is pictured in the right panel of Figure 5 for two-dimensional (that is, two-resource) systems: under

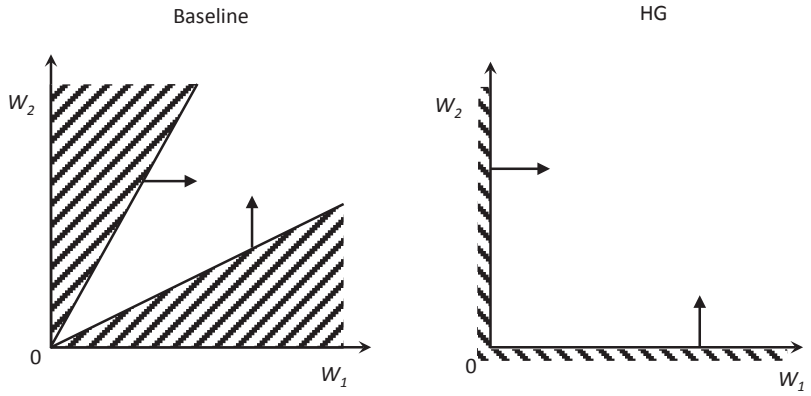

FIG 5. *Baseline versus HG workload state space.*

HG control the workload process W is reflected only at the boundary of the quadrant, which is interpreted to mean that capacity of each resource is fully utilized as long as there is any work in the system for that resource. The left panel of Figure 5 reproduces a figure from [14], showing the workload state space for the baseline diffusion process which those authors obtain as a heavy traffic limit under proportionally fair resource allocation; reflection occurs along two rays that lie strictly inside the non-negative quadrant, which corresponds to the occurrence of unused capacity under circumstances where it is avoidable.

11. Hierarchical greedy ideal (HGI) performance for the original model. In the foregoing discussion of Brownian approximations, we identified a hierarchical greedy control Y^* with associated performance measure $E[e \cdot Z^*(\infty)] = E[f(W^*(\infty))]$, where W^* is the minimum workload process defined by (10.3) and (10.4). For our original resource sharing network, we define an analogous *hierarchical greedy ideal* (HGI) performance goal as follows:

$$\text{HGI performance goal } = E[f(w^*(\infty))]. \tag{11.1}$$

where w^* is the minimum *actual* workload process defined via (7.5). Given the interpretation of $f(\cdot)$ that was provided in Section 8, it might seem more natural to define HGI performance in terms of nominal workload, rather than actual workload, but (a) as noted in Section 10, the distinction between nominal and actual workload is negligible in the heavy traffic parameter regime on which we focus, and (b) minimum actual workload is a well defined process in our original model setting, but minimum nominal workload is not.

TABLE 3
$E(Tot)$ comparison: Baseline versus HGI for our three examples

Example	Load Factor ρ	$E(Tot)$ Values Baseline	HGI	$\frac{\text{Baseline} - \text{HGI}}{\text{Baseline}}$
2LLN	0.80	8.00	5.89	26%
	0.90	18.00	12.75	29%
	0.95	38.00	26.61	30%
2LLN	0.80	12.00	8.80	27%
	0.90	27.00	19.06	29%
	0.95	57.00	39.46	31%
C3LN	0.80	12.00	8.80	27%
	0.90	27.00	16.00	41%
	0.95	57.00	33.29	42%

In broad terms, the obvious way for a system manager to pursue (11.1) as a performance goal is to use a hierarchical greedy approach akin to what was described in Section 10: first focus myopically (greedily) on workload minimization, or equivalently, on maximizing resource utilization (minimizing unused capacity); and then, given the constraints imposed by that dominant concern, strive to configure the backlog of work so that the associated cost rate is minimized. Our use of the word "ideal" in reference to (11.1) emphasizes the point that neither full resource utilization, nor cost rate minimization given workload, can be achieved *exactly* in a resource sharing network. On the other hand, the right-had side of (11.1) is *not* necessarily a lower bound on achievable performance, because of the non-monotonicity of $f(\cdot)$ that was demonstrated in Section 8. That is, a control policy that achieves HGI performance is not necessarily optimal, although one feels intuitively that (11.1) represents a high standard of performance.

To substantiate that view, Table 3 compares HGI and baseline values of $E(Tot)$ for our three examples, using three different load factors ρ: the $E(Tot)$ value in the column labeled "Baseline" is computed via formula (6.1), and the $E(Tot)$ value in the column labeled "HGI" is a simulation estimate of $E\left[f\left(w^*(\infty)\right)\right]$, in accordance with (11.1); see Appendix B for an explanation of the simulation logic. In these examples, HGI performance represents a 25–45% improvement over baseline performance, with greater percentage gains occurring at higher load factors. Also, the greatest percentage gains occur in the most complex of the three examples.

The analysis presented in Section 10 of this paper, when combined with arguments made in H2000 and earlier work referenced there, lead us to conjecture the existence of a control policy that achieves HGI performance in the heavy traffic limit; see Section 14 for elaboration. However, because the

Brownian system model provides such a highly compressed representation of the original resource sharing network, there is no obvious general way of translating achievable behavior in the Brownian model into an implementable control policy for the original system. That conundrum has been noted and discussed by various authors, starting with [9]. No general resolution has been found thus far, although [10] describes an approach using discrete-review policies that has broad potential applicability, and progress has been made for certain special network structures, such as the parallel server models studied by [2, 3].

12. Striving for HGI performance via UFOS. For the two-link linear network (2LLN) portrayed in Figure 1, it is possible to achieve the minimum workload vector $w^*(t)$ *exactly*, at every time $t \geq 0$ with probability 1, as follows. (a) Use one of the allocation vectors $x = (1, 1, 0)$ or $x = (0, 0, 1)$ whenever possible; the former choice keeps both resources fully utilized by processing type 1 jobs and type 2 jobs simultaneously, while the latter choice keeps both resources fully utilized by processing (only) type 3 jobs. (b) If only type 1 jobs are available for processing, choose $x = (1, 0, 0)$, and if only type 2 is available, choose $x = (0, 1, 0)$.

These rules are sufficient to achieve minimum workload (that is, they ensure that each resource will be fully utilized whenever there is work for it in the system), but they leave open the question of what to do when all three job types are present. In that case, the obvious choice for a system manager who wants to minimize $E(Tot)$ is $x = (1, 1, 0)$ rather than $x = (0, 0, 1)$, because the former decreases the total job count $n_1 + n_2 + n_3$ at an expected rate of $\mu_1 + \mu_2 = 2$, whereas the latter decreases $n_1 + n_2 + n_3$ at expected rate $\mu_3 = 1$. That is, when the primary criterion of maximizing resource utilization does not fully specify the control action, the remaining freedom should be used to decrease the instantaneous "cost rate" $n_1 + n_2 + n_3$ as rapidly as possible on an expected value basis. This scheme will be referred to by the acronym UFOS, which is mnemonic for *utilization first, output second*.

The UFOS allocation scheme is the obvious way to strive for HGI performance in the 2LLN In fact, [24] shows the following: if $\mu_1, \mu_2 \leq \mu_3$ and $\mu_3 \leq \mu_1 + \mu_2$ then UFOS stochastically minimizes $n_1(t) + n_2(t) + n_3(t)$ for each $t \geq 0$ in a two-link linear network, and hence it minimizes $E(Tot)$ as well The parameter values that we have assumed ($\mu_1 = \mu_2 = \mu_3 = 1$) satisfy those inequality constraints, so UFOS is exactly optimal for our 2LLN.

Moving now to the three-link linear network (3LLN) portrayed in Figure 2, the following analogous UFOS scheme immediately suggests itself: if all of job types 1, 2 and 3 are available for processing, choose the allocation vector

TABLE 4
$E(Tot)$ comparison: HGI versus UFOS for our three examples

Example	Load Factor ρ	$E(Tot)$ Values Baseline	HGI	$\frac{\text{UFOS} - \text{HGI}}{\text{HGI}}$
2LLN	0.80	5.89	6.05	3%
	0.90	12.75	13.26	4%
	0.95	26.61	27.13	2%
3LLN	0.80	8.80	8.94	2%
	0.90	19.06	19.81	4%
	0.95	39.46	40.47	3%
C3LN	0.80	8.80	8.12	-8%
	0.90	16.00	17.57	10%
	0.95	33.29	36.23	9%

$x = (1, 1, 1, 0)$; if one or more of those types is *not* available, but type 4 *is* available, choose $x = (0, 0, 0, 1)$; and if neither of those choices is available, allocate the capacity of each resource to local traffic in the obvious way. This UFOS scheme achieves the minimum workload vector $w^*(t)$ at every time t, and it maximizes the instantaneous output rate when doing so does not jeopardize full resource utilization, but it is *not* necessarily optimal, because with three links there exist system states where maximizing resource utilization requires some sacrifice in terms of the total output rate, and *vice versa*.

We conjecture that, for both the 2LLN and 3LLN, the UFOS allocation scheme described above will approach HGI performance asymptotically in the heavy traffic limit, by which we mean that the *percentage* difference between HGI performance and $E(Tot)$ under UFOS will vanish as $\rho \uparrow 1$. That conjecture is reasonably well supported by the simulation results reported in Table 4, where the percentage differences fall between 2% and 4% for all three values of ρ considered.

For the complex three-link network (C3LN) portrayed in Figure 3, the simulation results reported in Table 4 were derived using the following definition of UFOS: first, for any given state vector n, identify the set of allocation vectors $x = (x_j)$ that

$$\text{maximize } \sum_{i=1}^{I} (Ax)_i \text{ subject to } Ax \le C \text{ and } x \in \Phi(n), \tag{12.1}$$

where $\Phi(n)$ is defined by (2.1); and second, among the allocation vectors x that achieve the maximum in (12.1), choose one to

$$\text{maximize } \sum_{j=1}^{J} \mu_j x_j. \tag{12.2}$$

TABLE 5
Summary of $E(Tot)$ comparisons for our three examples

Example	Load Factor ρ	$E(Tot)$ Values: Baseline	PF Simulation	HGI	UFOS	$\frac{\text{PF} - \text{UFOS}}{\text{PF}}$
2LLN	0.80	8.00	7.33	5.89	6.05	17.5%
	0.90	18.00	17.17	12.75	13.26	22.8%
	0.95	38.00	37.11	26.61	27.13	26.9%
2LLN	0.80	12.00	10.56	8.80	8.94	15.3%
	0.90	27.00	25.2	19.06	19.81	21.4%
	0.95	57.00	53.97	39.46	40.47	25.0%
C3LN	0.80	12.00	10.34	8.80	8.12	21.5%
	0.90	27.00	24.75	16.00	17.57	29.0%
	0.95	57.00	54.29	33.29	36.23	33.3%

The objective function in our first-level optimization (12.1) is simply the sum of the utilization rates for the various resources, and the equal weighting used in that objective function is arbitrary: any weighted sum of the utilization rates having all weights strictly positive would be consistent with our earlier specification of UFOS for the 2LLN and 3LLN.

A striking feature of Table 4 is that, for the C3LN with load factor $\rho = 0.80$, the simulation estimate of $E(Tot)$ under UFOS is actually 8% *below* HGI performance, which underscores the point that HGI performance is not necessarily a bound on optimal performance (see Section 11). At higher load factors, the ordering of UFOS performance and HGI performance is reversed, and the gap between them is substantial. There is no compelling reason to believe that the percentage gap between HGI and UFOS will vanish as $\rho \uparrow 1$, but we have no better scheme to recommend as a means of approaching HGI performance in the C3LN.

13. Recap and a negative example. Table 5 combines results presented earlier (specifically, in Tables 1, 3 and 4), in order to underscore the following points about our three examples. First, the baseline formula (6.1) closely approximates simulated performance under proportional fairness (PF). Second, the hierarchical greedy ideal (HGI) formula (11.1) represents a 25–45% improvement relative to baseline. And third, the UFOS allocation scheme defined by (12.1) and (12.2) gives $E(Tot)$ values reasonably close to HGI performance for these examples.

In particular, as shown in the right-most column of Table 5, $E(Tot)$ values under UFOS are 15–35% lower than those under proportional fairness, with greater relative improvements achieved at higher load factors. These are moderate but significant performance gains. We conjecture that similar gains would be achievable, relative to the performance of proportional fair-

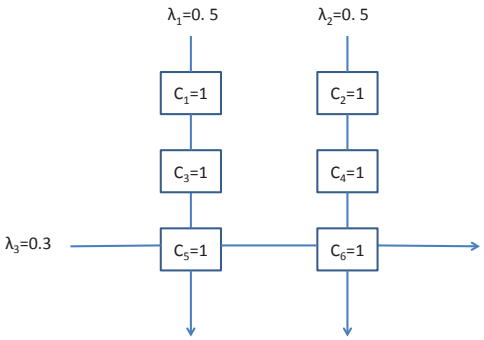

FIG 6. *An example where UFOS fails.*

ness and other frequently cited allocation schemes, if a congestion measure different from $E(Tot)$ were used. For example, if the objective were to minimize a quadratic function of the steady-state job counts, rather than the linear function embodied in $E(Tot)$, the second-stage logic of UFOS could be altered to drive down that quadratic cost as quickly as possible on an expected value basis.

Unfortunately, the UFOS allocation scheme (as we have defined it) is not generally effective. This is illustrated by the example portrayed in Figure 6, which was suggested by R. Srikant. (This example does not satisfy our local traffic assumption, but one can add to it a stream of local traffic for each resource, with each such stream having mean job size 1 and an average arrival rate of, say, $\rho/100$. The local traffic will then constitute an insignificant fraction of the load on any given resource, and everything said here will remain essentially the same.) In Figure 6 there are three job types, and the mean job size is assumed to be 1 for each of them. As shown on the figure, each of the six resources has capacity 1. Average arrival rates are as shown on the figure, so resources 1 through 4 all have a load factor of 0.5, while resources 5 and 6 each have a load factor of 0.8.

Job types 1 and 2 both utilize three resources, whereas type 3 utilizes only two resources. Thus the first-stage UFOS optimization (12.1) effectively gives priority to types 1 and 2: if there are any jobs of either type 1 or type 2 present in the system, a flow rate of 1 will be allocated to them, while type 3 is given a zero allocation. Because of their priority status, jobs of types 1 and 2 both enter what is effectively an M/M/1 queue with load factor 0.5, and those two M/M/1 queues operate independently of one another; the steady-state probability that either n_1 or n_2 individually equals zero is 0.5, and the steady-state probability that $n_1 = n_2 = 0$ is 0.25. Type 3 jobs

receive a flow rate allocation of 1 when $n_1 = n_2 = 0$, and an allocation of zero otherwise, which is inadequate to meet the type 3 input rate of 0.3. Thus the system portrayed in Figure 6 is actually unstable under UFOS as we have defined it, even though no resource has a load factor larger than 0.8.

The problem, of course, is that UFOS is "distracted" by the lightly loaded resources numbered 1 through 4, giving just as much weight to keeping them busy as to keeping busy the critical resources numbered 5 and 6. One approach to fixing that problem is to use a *weighted* sum of utilization rates as the objective function in (12.1), with larger weights for more heavily loaded resources, or to adjust the weights dynamically depending on the current workloads of different resources. While that idea is attractive in principle, we have not been able to devise a provably effective implementation thus far.

14. Open problem. In conclusion, it may be helpful to state more precisely the "open problem" referred to in the title of this paper. Consider a general resource sharing network of the kind described in Section 2, with $\rho_i < 1$ for each resource i, viewing the arrival rates $\lambda_1, \ldots, \lambda_J$ as variable parameters. Initially, consider a sequence of values for the arrival rate vector such that $\rho_i \uparrow 1$ for each resource i, and moreover, $(1 - \rho_i)/(1 - \rho_k)$ converges to a strictly positive constant for every pair of resources i and k (that is, the load factors for different resources converge to 1 at the same rate). This type of formulation, in which all resources are equally "critical," is more or less standard in heavy traffic theory. Using the standard approach to asymptotic optimality, one would then state the problem as follows: develop a corresponding sequence of dynamic allocation schemes such that the percentage difference between the $E(Tot)$ values they achieve and HGI performance vanishes. Presumably that would be accomplished by constructing controls whose associated job count processes, properly scaled, converge weakly to the process Z^* defined in Section 7.

A more stringent version of the problem would require that the dynamic allocation logic *not depend on arrival rates*; in the literature of communication networks, this is commonly cited as a desirable characteristic, because arrival rates may vary through time and one wants a control scheme that remains effective in the face of such changes. With that constraint, the problem is effectively to find *a single dynamic allocation scheme* which, when applied with the specified sequence of arrival rate vectors, causes the percentage difference between achieved $E(Tot)$ values and HGI performance to vanish. Of course, one wants a dynamic allocation scheme which has that property for *any* sequence of arrival rate vectors that take the system into heavy traffic.

Finally, it is of interest to consider the broader heavy traffic regime where $\rho_i \uparrow 1$ for some resources i but not necessarily for all of them, or where the

load factors for different resources approach 1 at different rates. Again the problem is to formulate a control policy such that the percentage difference between HGI performance and the policy's achieved $E(Tot)$ value vanishes in the limit. With this broadened view of "heavy traffic," there arises the distinction between "critical" and "sub-critical" resources: the steady-state minimum workloads for sub-critical resources are eventually insignificant compared to the steady-state minimum workloads for critical ones, and so sub-critical resources are eventually irrelevant for computing HGI performance values. Still, the preceding discussion of Srikant's example (Figure 6) shows that the potential presence of sub-critical resources substantially complicates the task of designing general control logic.

APPENDIX A: MORE ON WORKLOAD PROCESSES

We shall consider in this appendix an ordinary M/M/1 queueing system, which can be viewed as a special case of the resource sharing network described in Section 2. To be specific, it is the special case where $I = J = 1$ and the capacity consumption matrix A consists of a single 1. Two results will be developed in that simplified setting, and then their obvious analogs for general networks will be stated.

Consider an M/M/1 system with service rate $\mu = 1$, arrival rate $\lambda < 1$, and initial state $n(0) = 0$ We view λ as a variable parameter, define $\varepsilon = \sqrt{1-\lambda}$, and eventually consider the heavy traffic limit where $\varepsilon \downarrow 0$. Assuming that server capacity is allocated to jobs in a work conserving manner that does not depend on the jobs' service times (for example, it could be FIFO, LIFO or processor sharing) the memoryless property of exponential service times gives us the following:

$$w(t) \sim S(n(t)) \text{ for each fixed } t > 0, \tag{A.1}$$

where $\{n(t), t \geq 0\}$ is the job count process as in Section 2, $\{w(t), t \geq 0\}$ is the actual workload process as in Section 7, "$\sim$" denotes equivalence in distribution, $S_k = \eta_1 + \cdots + \eta_k$ for $k = 1, 2, \ldots$, and $\eta_1, \ldots, \eta_k$ are independent, exponentially distributed random variables with mean 1, also independent of $n(t)$. Because the mean service time (mean job size) is 1 by assumption, the nominal workload process $\{\hat{w}(t), t \geq 0\}$ defined in Section 7 is simply

$$\hat{w}(t) = n(t), \ t \geq 0. \tag{A.2}$$

Using the symbol "$\Rightarrow$" to denote convergence in distribution, we have $n(t) \Rightarrow n(\infty)$ as $t \to \infty$, where $n(\infty)$ has a specific distribution that need not concern us here. From that and (A.1) it follows (using the continuity

theorem for Laplace transforms, for example) that

$$w(t) \Rightarrow w(\infty) \text{ as } t \to \infty, \text{ where } w(\infty) \sim S(n(\infty)). \tag{A.3}$$

If one considers a resource sharing network operating under any stable control, we have $n(t) \Rightarrow n(\infty)$ as a matter of definition, and then one obtains $w(t) \Rightarrow w(\infty)$ by precisely similar reasoning. That is, there exists a limiting actual workload distribution under any stable control strategy, as claimed in Section 7.

To compare the nominal and actual workload processes in heavy traffic, it will be useful to define $\hat{S}(k) = S(k) - k$ for $k = 1, 2, \ldots$, observing that $\hat{S}(\cdot)$ is the partial sums process for an independent and identically distributed sequence with zero mean. From (A.1) and (A.2) we have

$$w(t) - \hat{w}(t) \sim \hat{S}\,(n(t)) \text{ for each fixed } t > 0. \tag{A.4}$$

Let us now consider the parametric family of M/M/1 systems with $\varepsilon \downarrow 0$, using the notation $n^{\varepsilon}(t)$ to indicate explicitly the dependence of the job count process on the parameter ε, and similarly for $w^{\varepsilon}(t)$ and $\hat{w}^{\varepsilon}(t)$. For the diffusion scaled job count process we have

$$\varepsilon n^{\varepsilon}(\varepsilon^{-2}t) \Rightarrow Z(t) \text{ as } \varepsilon \downarrow 0 \text{ for each fixed } t > 0, \tag{A.5}$$

where$\{Z(t), t \geq 0\}$ is a certain one-dimensional reflected Brownian motion. This is the standard heavy traffic limit theorem for an M/M/1 system; see, for example, Section 6.4 of [7]. A bit more work gives

$$E\left[\varepsilon n^{\varepsilon}(\varepsilon^{-2}t)\right] \Rightarrow E[Z(t)] < \infty \text{ as } \varepsilon \downarrow 0 \text{ for each fixed } t > 0 \tag{A.6}$$

as well. Let us now consider the difference between the diffusion scaled actual and nominal workload processes, defining

$$\Delta^{\varepsilon}(t) = \varepsilon w(\varepsilon^{-2}t) - \varepsilon \hat{w}(\varepsilon^{-2}t), \ t \geq 0. \tag{A.7}$$

Denoting by σ^2 the variance of the service time (job size) random variables $\{\eta_k\}$, we have from (A.4) that

$$E[\Delta^{\varepsilon}(t)] = 0 \text{ and Var } [\Delta^{\varepsilon}(t)] = \varepsilon^2\sigma^2 E[n^{\varepsilon}(\varepsilon^{-2}t)] = \varepsilon\sigma^2 E[\varepsilon n^{\varepsilon}(\varepsilon^{-2}t)]. \tag{A.8}$$

Combining this with (A.6), we conclude that $\Delta^{\varepsilon}(t) \to 0$ in the L^2 sense, and hence also in probability, as $\varepsilon \downarrow 0$ for fixed $t > 0$, and the argument is easily extended to give $\Delta^{\varepsilon} \Rightarrow 0$ in the functional sense.

An argument virtually identical to that in the previous paragraph establishes a similar conclusion for general resources sharing networks. That is, if we consider a family of such networks parameterized by $\varepsilon > 0$, operating under a stable control such that (A.5) and (A.6) hold for some limit process Z, then the difference between the diffusion scaled nominal and actual workload processes vanishes in the heavy traffic limit, as claimed in Section 10.

APPENDIX B: ESTIMATING HGI PERFORMANCE VIA MONTE CARLO SIMULATION

Estimates of the HGI performance goal $E[f(w^*(\infty))]$ were obtained, for each of our three network examples and for each of the ρ values considered in Section 11, using a two-step procedure. The first step generates a sample path of the minimum workload process w^* defined in Section 7, as follows. Poisson arrivals and exponentially distributed job sizes are generated for each job type j, from which we construct sample paths of the workload input processes $\{\ell_i(t), t \geq 0\}$ that were defined verbally in Section 7: the sample path of $\ell_i(\cdot)$ for a given resource i starts at $\ell_i(0) = 0$, is constant between arrival epochs, and jumps upward by an amount $A_{ij}S$ when a type j job arrives and that job has size S. Next, the sample path of the minimum workload process $\{w_i^*(t),\ t \geq 0\}$ is constructed independently for each resource i, exactly as one constructs the content process of a dam with cumulative input process $\ell_i(\cdot)$ and constant outflow rate C_i. That is, the sample path of $w_i^*(\cdot)$ starts at $w_i^*(0) = 0$, has upward jumps identical to those of $\ell_i(\cdot)$, slopes downward at rate C_i until $w_i^*(\cdot) = 0$ again, and then remains at zero until the next jump of $\ell_i(\cdot)$ occurs. A *regenerative cycle* is completed at the first time τ, following the first arrival of a job of any type, when we once again have $w_i^*(\tau) = 0$ for all $i = 1, \ldots, I$.

The second step in our estimation procedure is to calculate, given a piecewise linear sample path $\{w^*(t),\ 0 \leq t \leq \tau\}$ of the vector process w^* over a regenerative cycle, the integral

$$F = \int_0^\tau f(w^*(t))dt. \tag{B.1}$$

Let $0 = T_0 < \cdots < T_K = \tau$ be a sequence of times such that all components of $w^*(\cdot)$ are linear over each sub-interval $[T_{k-1}, T_k), k = 1, \ldots, K$. That is, each break point T_k is either the arrival time of some job or else a time at which some component of w^* hits zero from above. In all three of our examples we have $C_i = 1$ for each resource i, which implies the following: over each of the subintervals $[T_{k-1}, T_k)$, each component $w_i^*(\cdot)$ of the minimum workload process is either identically zero or else linear with slope -1. It follows from that special structure and the definition of $f(\cdot)$ that $f\left(w^*(t)\right)$ is itself linear in t over each of the subintervals $[T_{k-1}, T_k)$, implying that

$$\int_{T_{k-1}}^{T_k} f\left(w^*\left(t\right)\right) dt = \frac{1}{2}\left[f\left(w^*(T_{k-1}+)\right) + f(w^*\left(T_k-\right))\right](T_k - T_{k-1}). \tag{B.2}$$

Thus, exact computation of the integral in (B.2) requires only that $f(\cdot)$ be evaluated for finitely many values of its argument, which is easily done:

explicit formulas for $f(\cdot)$ were given in Section 7 for our 2LLN and 3LLN, and a similar, more complicated formula was developed for the C3LN.

We generated a large number N of regenerative cycles, recording their durations $\tau_1, \ldots, \tau_N$ and the corresponding values $F_1, \ldots, F_N$ for the integral in (B.2). Using the regenerative method, we then estimated the HGI performance goal as

$$E\left[f(w^*(\infty))\right] \simeq (F_1 + \cdots + F_N)/(\tau_1 + \cdots + \tau_N).$$

Acknowledgments. An early version of the research reported here was presented at the MIT Stochastic Networks Conference in June of 2012. Laurent Massoulié asked whether UFOS is a stable control whenever the traffic condition (4) is satisfied, and a few days later R. Srikant provided the negative example described in Section 12, albeit in a somewhat different form. We have further benefited from conversations with Baris Ata, Sem Borst, Jim Dai, Dan Iancu, Balaji Prabhakar, and Ruth Williams, and from the comments and criticisms of two anonymous referees.

REFERENCES

[1] Baskett, F., Chandy, K. M., Muntz, R. R. and Palacios, J. (1975). Open, closed, and mixed networks of queues with different classes of customers. *J. of the ACM*, **22** 248–260. MR0365749

[2] Bell, S. L. and Williams, R. J. (2001). Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: Asymptotic optimality of a threshold policy. *Ann. Appl. Probab.*, **11** 608–649. MR1865018

[3] Bell, S. L. and Williams, R. J. (2005). Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: Asymptotic optimality of a threshold policy. *Elect. J. Probab.*, **10** 1044–1115. MR2164040

[4] Bonald, T. and Proutiére, A. (2002). Insensitivity in processor-sharing networks. *Performance Evaluation*, **49** 193–209.

[5] Bonald, T. and Proutiére, A. (2003). Insensitive bandwidth sharing in data networks. *Queueing Systems*, **44** 69–100. MR1989867

[6] Bramson, M. (2008). *Stability of Queueing Networks*. Springer, New York. MR2445100

[7] Chen, H. and Yao, D. D. (2001). *Fundamentals of Queueing Networks*. Springer, NewYork. MR1835969

[8] de Veciana, G., Lee, T.-J. and Konstantopoulos, T. (2001). Stability and performance analysis of networks supporting elastic services. *IEEE/ACM Trans. on Networking*, **9** 2–14

[9] Harrison, J. M. (1988). Brownian models of queueing networks with heterogeneous customer populations, in W. Fleming and P.-L. Lions (eds.), *Stochastic Differential Systems, Stochastic Control Theory and Applications*, IMA Volumes in Mathematics and Its Applications, **10** 147–186. Springer-Verlag, New York. MR0934722

[10] Harrison, J. M. (1996). The BIGSTEP approach to flow management in stochastic processing networks, in F. P. Kelly, S. Zachary, and I. Ziedins (eds.), *Stochastic Networks: Theory and Applications*, 57–90. Oxford University Press.

[11] Harrison, J. M. (2000). Brownian models of open processing networks: Canonical representation of workload. *Ann. Appl. Probab.*, **10** 75–103. Correction **13** (2003) 390–393. MR1765204

[12] Harrison, J. M. (2013). *Brownian Models of Performance and Control*. Cambridge University Press, New York. MR3157450

[13] Harrison, J. M. and J. A. Van Mieghem (1997). Dynamic control of Brownian networks: State space collapse and equivalent workload formulations. *Ann. Appl. Probab.*, **7** 747–771. MR1459269

[14] Kang, W. N., Kelly, F. P., Lee, N. H. and Williams, R. J. (2009). State space collapse and diffusion approximation for a network operating under a fair bandwidth sharing policy. *Ann. Appl. Probab.*, **19** 1719–1780. MR2569806

[15] Kelly, F. P. (1979). *Reversibility and Stochastic Networks*. John Wiley and Sons, New York. MR0554920

[16] Kelly, F. P. (1997). Charging and rate control for elastic traffic. *Eur. Trans. Telecom.*, **8** 33–37.

[17] Kelly, F. P., Maulloo, A. and Tan, D. (1998). Rate control for communication networks: Shadow price, proportional fairness and stability. *J. Oper. Res. Soc.*, **49** 237–252.

[18] Kelly, F. P. and Williams, R. J. (2004). Fluid model for a network operating under a fair bandwidth-sharing policy. *Ann. Appl. Probab.*, **14** 1055–1083. MR2071416

[19] Kenyon, T. (2002). *High-Performance Data Network Design*. Digital Press. Boston.

[20] Massoulié, L. and Roberts, J. (2000). Bandwidth sharing and admission control for elastic traffic. *Telecommunication Systems*, **15** 185–201.

[21] Mo, J. and Walrand, J. (2000). Fair end-to-end window-based congestion control. *IEEE/ACM Trans. on Networking*, **8** 856–567.

[22] Shah, D., Tsitsiklis, J. N. and Zhong, Y. (2013). Qualitative properties of α-fair policies in bandwidth sharing networks. *Ann. Appl. Probab.*, **24** 76–113. MR3161642

[23] Verloop, I. M., Borst, S.C. and Núñez Queija, R. (2005). Stability of size-based scheduling disciplines in resource-sharing networks. *Performance Evaluation*, **62** 247–262.

[24] Verloop, I. M. and Núñez Queija, R. (2009). Assessing the efficiency of resource allocations in bandwidth-sharing networks. *Performance Evaluation*, **66** 59–77.

[25] Walkup, D. W. and Wets, R. J.-B. (1969). Lifting projections of convex polyhedra. *Pacific J. of Math.*, **28** 465–475. MR0242055

[26] Zachary, S. (2007). A note on insensitivity in stochastic networks. *J. of Appl. Prob.*, **44** 238–248. MR2312999

Graduate School of Business
Stanford University
Stanford, CA 94305

Department of Electrical Engineering
Stanford University
Stanford, CA 94305

Department of Electrical Engineering
and Computer Science
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139