WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

SCHEDULING A TWO-STATION MULTICLASS QUEUEING NETWORK IN HEAVY TRAFFIC

Lawrence M. Wein

Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

#2016-88

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

SCHEDULING A TWO-STATION MULTICLASS QUEUEING NETWORK IN HEAVY TRAFFIC

Lawrence M. Wein

Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

#2016-88

SCHEDULING A TWO-STATION MULTICLASS QUEUEING NETWORK IN HEAVY TRAFFIC

Lawrence M. Wein

Abstract

Motivated by a factory scheduling problem, we consider the problem of input control (subject to a specified product mix) and sequencing in a two-station multiclass queueing network with general service time distributions and a general routing structure. The objective is to minimize the long-run average expected number of customers in the system subject to a constraint on the long-run average expected output rate. Under balanced heavy loading conditions, this scheduling problem is approximated by a control problem involving Brownian motion. A reformulation of this Brownian control problem was solved exactly in Wein [17]. In the present paper, this solution is interpreted in terms of the queueing network model in order to obtain an effective scheduling rule. The resulting sequencing rule is a static priority ranking of the classes. The input policy is a "workload regulating" input policy, where a customer is injected into the system whenever the expected total amount of work in the system for the two stations falls within a prescribed region. An example is presented that illustrates the procedure and demonstrates its effectiveness.

November 1987

SCHEDULING A TWO-STATION MULTICLASS QUEUEING NETWORK IN HEAVY TRAFFIC

Lawrence M. Wein

This research is motivated by a particular scheduling problem that is encountered in many factories. By viewing a factory as a network of queues, the scheduling problem can be formulated as one of controlling the flow in a queueing network. The queueing network under consideration consists of two single-server stations and K different customer classes. Customers of class $k = 1, ..., K$ require service at a specific station $s(k)$ and their service times are independent and identically distributed random variables with finite mean m_k and variance s_k^2. Upon completion of service, a class k customer turns next into a class j customer with probability P_{kj} and exits the system with probability $1 - \sum_{j=1}^{K} P_{kj}$, independent of all previous history. We assume that the $K \times K$ Markovian switching matrix $P = (P_{kj})$ has spectral radius less than one, so that all customers will eventually exit the system. Because the number of classes is allowed to be arbitrary, this routing structure is almost perfectly general.

The scheduling problem incorporates input and sequencing decisions. We assume there is an endless line of customers who are waiting to gain entry into the network. Each customer in the line has an exogenously specified class designation. These class designations are such that, over the long-run, the proportion of class k customers released into the system is q_k, where $\sum_{k=1}^{K} q_k = 1$. The vector $q = (q_k)$ will be referred to as the *entering class mix*. The input decisions are to choose a non-decreasing process $N = \{N(t), t \geq 0\}$, where $N(t)$ is the cumulative number of customers injected into the system up to time t. Thus the input decisions essentially allow full discretion over the *timing* of the release of customers into the system, but do not allow for the choice of which

class of customer to inject.

The sequencing decisions consist of choosing, at each point in time, which *class* of customer to process at each server in the network. Preemptive resume scheduling is allowed, so that service of a customer may be interrupted at a particular station when a higher priority customer arrives at that station. Due to the rather crude nature of the Brownian approximation that is employed here, the assumptions made regarding preemption do not have an effect on the scheduling policy that emerges from the analysis.

It is assumed that a holding cost c_k is incurred for each unit of time that a class k customer spends in the queueing network. Also, there is a specified lower bound $\bar{\lambda}$ on the long-run average expected *throughput rate* of the queueing network. The throughput rate of a queueing system is the number of customer departures from the system per unit of time. Our queueing network scheduling problem is to choose the input and sequencing decisions so as to minimize the long-run average expected holding costs incurred per unit of time, subject to a lower bound constraint on the long-run average expected throughput rate. Notice that in the special case where $c_k = c$ for all $k = 1, ..., K$, the objective is to minimize the long-run average number of customers in the system. Because the problem is formulated in terms of long-run averages and because the constraint on throughput will in general be tight, Little's formula [9] implies that this objective is equivalent to minimizing the long-run average expected *cycle time* of customers in the system. The cycle time of a customer is the amount of time a customer spends in the queueing network. In a manufacturing setting, there are many good reasons to minimize both the work-in-process inventory and the cycle time, and some of these will be discussed in the next section.

A good deal of literature exists on input control of queueing networks, but these models consider the decision of whether to accept or reject Poisson arrivals; Stidham [15] provides a thorough survey of work in this area. Such models are not applicable to the scheduling problem considered here, since the relevant issue in our setting is *when* to release a customer into the queueing network, not whether or not to accept the customer. Although useful

results exist for sequencing single-station systems (see Klimov [8]), a satisfactory theory for sequencing in a network setting has not been attained, and simulation (see Conway, Maxwell and Miller [2] for a classic study on this topic) is still the primary tool of analysis. In view of the difficulty in obtaining sequencing rules for conventional multiclass queueing networks (it has been 14 years since Klimov's result), the best hope for further progress appears to be in the analysis of cruder, more tractable models.

One such model is a Brownian network, a stochastic system model introduced by Harrison [4]. Under conditions of *balanced heavy loading*, a Brownian network approximates a multiclass queueing network with dynamic scheduling capability. To state these conditions more precisely, let the two-vector $\rho = (\rho_i)$ be the relative server utilizations, or *traffic intensities*, for the two stations. The values of ρ_1 and ρ_2 can be computed from the switching matrix P, the vector $m = (m_k)$ of expected processing times, the entering class mix $q = (q_k)$ and the specified average throughput rate $\bar{\lambda}$, as will be shown in Section 2. The balanced heavy loading conditions assume the existence of a large integer n such that $0 \leq \sqrt{n}(1-\rho_i) \leq 1$ for $i = 1, 2$. As a canonical example, one may think of $\rho_1 = \rho_2 = .9$, in which case $n = 100$ satisfies this condition.

Under such conditions, the scheduling problem described above can be approximated by a dynamic control problem for a Brownian network. The state of the system in this Brownian control problem is a K-dimensional vector queue length process (appropriately scaled). Instead of analyzing the Brownian control problem directly, the problem is reformulated in Wein [17] so that the state of the system is described by a two-dimensional process that represents the scaled version of the total amount of work in the system for each of the two stations. The reformulated problem is solved exactly in Wein [17], and in the present paper, the solution is interpreted in terms of the queueing system in order to obtain an effective scheduling rule for the original queueing network (and hence factory) scheduling problem. This interpretation is based on intuition obtained from existing heavy traffic limit theorems for some simpler queueing systems, and no attempt is made to rig-

orously justify our interpretation via a weak convergence result. However, we conjecture that the resulting scheduling rule is asymptotically optimal in the heavy traffic limit (i.e., as $n \to \infty$).

The scheduling rule derived here consists of a sequencing rule and an input policy. To describe the rule, a few definitions are needed. Let M_{ik} equal the expected total amount of time that the server at station i (hereafter referred to as server i) must devote to a class k customer before that customer eventually exits the network. Denote the K-dimensional queue length process by Q, so that $Q_k(t)$ is the number of class k customers in the system at time t for $k = 1, ..., K$. Defining a two-dimensional *workload process* $w = (w_i)$ by $w(t) = MQ(t)$, where $M = (M_{ik})$, we interpret $w_i(t)$ as the expected total amount of work for server i embodied in those customers who are present *anywhere in the network* at time t.

Recalling that c_k is the linear holding cost for a class k customer, the sequencing rule ranks each customer class k by the index $c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k})$. In the special case where $c_k = c$ for all $k = 1, ..., K$, this rule is a static priority ranking that awards higher priority at station 1 (respectively, station 2) to the smaller (respectively, larger) values of this index. (The case where $c_k \neq c$ for all $k = 1, ..., K$ will be discussed in Section 5). It is interesting to note that, as in Klimov's results for a single-station queueing system, the solution to a *dynamic* scheduling problem is a *static* priority ranking of the classes, and the solution depends on the general service time distributions only through their means.

This sequencing rule has the following interpretation. In the special case when $c_k = c$ for all $k = 1, ..., K$ and $\rho_1 = \rho_2$ (i.e., minimization of the cycle time in a perfectly balanced system), the rule tends to retain jobs at each station (by giving them lower priority) that have relatively more work to be done at that station, either now or later, dispatching more quickly (by giving them higher priority) jobs that have relatively more work to be done at the other station. When $\rho_1 \neq \rho_2$, then ρ_1 and ρ_2 show up as appropriate weighting factors. Incidentally, it is known (Harrison and Wein [5]) that when $c_k = c$ for all $k = 1, ..., K$,

this same sequencing rule maximizes the throughput rate in a two-station multiclass *closed* queueing network in heavy traffic.

The input rule is called a *workload regulating policy* because it depends solely on the two-dimensional workload process w. More specifically, the rule releases a new customer into the system whenever the workload process enters a certain region in the nonnegative orthant of $\mathbf{R^2}$. A description of this region is fairly involved and will be deferred until Section 6, where the region is calculated explicitly. For a typical example, interested readers may refer to Figure 2 of Section 7, where the region consists of the shaded area. The input rule causes the network to behave as a "pull" system: when either server appears to be threatened with idleness *and* there is not too much work already present in the system, a new customer is released into the system.

Although neither the sequencing nor input rule derived here has ever appeared in the literature, they are both intuitively appealing policies. Furthermore, in a manufacturing setting, they would be very easy to implement. As will be seen in Section 7, these policies outperform conventional scheduling rules in simulation studies.

The original system description of a two-station, heavily-loaded, well-balanced network may seem quite restrictive at first glance. However, one important implication of the balanced heavy loading assumption is that, in the heavy traffic limit represented by the Brownian network model, any stations in the original system that are not among the most heavily loaded will simply disappear. This has been proven in limit theorems by Johnson [7] and Chen and Mandelbaum [1] in the single-type open queueing network setting. Limit theorems of this type can justify the procedure of eliminating all stations that are not heavily loaded when forming the approximating Brownian network, reducing the original system to a subnetwork of bottleneck stations for purposes of subsequent analysis. However, these bottleneck stations are *precisely* where the large queues form, where most of the waiting is incurred, and thus where scheduling will have the biggest impact. In fact, other approaches to job shop scheduling problems, such as the the OPT system (see

Jacobs [6] for a critical evaluation of its main features) or the expert systems approach taken by Morton and Smunt [10], also focus on the bottleneck stations. Thus, although the Brownian network approximation is a rather crude model in comparison to a conventional queueing network, its underlying assumptions are made-to-order for scheduling purposes.

One consequence of the previous paragraph is that the scheduling rule emerging from our analysis can be applied to *any queueing network with two bottleneck stations.* In fact, a simulation model has been built that is based on operating data from an actual semiconductor wafer fabrication facility. Using this simulation model (see Wein [16] for details), which contains 24 stations but only two bottleneck stations, rules similar to the ones derived here were compared against conventional sequencing and input rules. The results were quite impressive: the rules outperformed conventional rules and achieved a 47.2% reduction in average customer queueing time versus the base case of Poisson inputs and first-in first-out (FIFO) sequencing.

This paper is organized as follows. The factory scheduling problem that motivates our study is discussed in Section 1. In Section 2 the Brownian approximation of the queueing network scheduling problem is stated. The Brownian control problem is reformulated in Section 3 and the solution to the reformulated problem, which was derived in Wein [17], is stated in Section 4. This solution is interpreted in terms of the original queueing system in Sections 5 and 6, in order to obtain a sequencing rule and an input policy, respectively. An example is presented in Section 7 that illustrates the procedure and demonstrates its effectiveness.

Some of the notational conventions and terminology used in this paper will now be introduced. A stochastic process is said to be RCLL if its sample paths are right continuous and have left limits with probability one. When we say that X is a (μ, σ^2) Brownian motion, it is assumed there is a given $(\Omega, \mathbf{F}, \mathbf{F_t}, X, P_x)$, where $(\Omega, \mathbf{F})$ is a measurable space, $X = X(\omega)$ is a measurable mapping of Ω into $\mathbf{C}(\mathbf{R})$, which is the space of continuous functions on the real line $\mathbf{R}$, $\mathbf{F_t} = \sigma(X(s), s \leq t)$ is the filtration generated by X, and P_x

is a family of probability measures on Ω such that the process $\{X(t), t \geq 0\}$ is a Brownian motion with drift μ, variance σ^2 and initial state x. Let E_x be the expectation operator associated with P_x. If $Y = \{Y(t), t \geq 0\}$ is a process that is $\mathbf{F_t}$-measurable for all $t \geq 0$, then we say that the process Y is *non-anticipating* with respect to the Brownian motion X. More generally, we will say that one process Y is non-anticipating with respect to another process X when Y is adapted to the coarsest filtration with respect to which X is adapted.

1. The Factory Scheduling Problem

This section describes the relationship between the queueing network scheduling problem and the factory scheduling problem. Each server in the queueing network corresponds to a machine or work center in the factory, and each customer corresponds to a particular job. The routing structure described in the introduction can accomodate the case where the factory produces a variety of products, each with its own arbitrary deterministic route through the network of machines. In that case, a different customer class is defined for each *combination* of product and stage of completion. More generally, our set-up allows probabilistic routing to represent such events as rework or scrapping. In fact, a customer class can include any observable information about a particular job that is relevant for dynamic scheduling purposes.

The queueing network model can also accomodate machine breakdown and repair. By assuming that the amount of machine busy time between consecutive breakdowns is exponentially distributed, the breakdown and repair can be incorporated into the service time distributions for each customer class; see Harrison [4] for details. The modified m_k and s_k^2 are interpreted as the mean and variance of the *effective* service time of a class k customer, i.e., the actual processing time plus the total duration of all interruptions that

occur during that service.

In the manufacturing setting, the sequencing decisions consist of dynamically choosing which job to process at each machine in the factory; this corresponds to the classic job shop scheduling problem. The input decisions in our problem specify the timing of the release of jobs onto the factory floor. However, it is assumed that the exact sequence of entering product types is specified precisely. This sequence reflects the desired product mix that the factory is required to maintain. For example, if a factory makes two products, A and B, to be produced in equal quantities, then the specified sequence of entering jobs would be ABABAB... There is a long-run average output rate (in jobs per unit time) that the factory is required to maintain. When the holding costs $c_k = c$ for all $k = 1, ..., K$, the objective is to minimize the long-run average expected work-in-process (WIP) inventory, which is equivalent to minimizing the long-run average expected cycle time of jobs.

This scheduling problem is relevant for any factory that is obliged to maintain a specified average output rate of a certain product mix, but can control the timing of its inputs. In thinking about endogenously generated arrivals, it is easiest to imagine a make-to-stock manufacturer, where orders are met from finished goods inventory. However, in a make-to-order environment, input to the factory floor can also be regulated, but then customer orders will sometimes queue outside the factory floor waiting to gain entrance. The motivation for doing this is to reap the benefits that can be gained by a reduction in both the WIP inventory on the factory floor and the cycle time of jobs on the factory floor. By reducing the number of jobs on the factory floor, the benefits from *Just-In-Time* manufacturing (see Schonberger [13] for a detailed description) can be realized. For example, quality problems will be detected faster, and thus there will be less rework and scrap of jobs. By reducing the cycle time of jobs, the factory can gain *flexibility*: the system will be more capable of very fast turnaround on individual orders, and the factory may more readily adapt to a changed order, since the corresponding job may not have begun its processing. A more specific example occurs in the semiconductor industry, where a

decrease in the average cycle time of a lot of wafers in the wafer fab will result in an increase in the yield of good wafers. This is because lots are so easily contaminated while in the fab. Finally, in the case of standardized products that can be made to stock, shorter cycle times allow production to be based on more accurate forecasts of market demand.

Since our definition of cycle time does not include the time that transpires between receiving an individual order and releasing the corresponding job onto the factory floor, readers may be concerned about the effect the rules derived here would have on due-date performance. Our view is that, in the case of a busy factory with more than one bottleneck machine, scheduling for due-dates has a detrimental effect on the utilization of bottleneck machines, and hence ultimately does more harm than good. As an example, consider a two-station well-balanced factory that has a very large backlog of jobs, each with a given due date. Furthermore, suppose that either workload regulating input (see Section 6) or closed loop input (the total number of jobs on the factory floor is held constant, see Solberg [14]) is used. In these cases, the sequencing rule described here can substantially increase server utilization compared to *any* sequencing rule that sequences according to due-date information (see Harrison and Wein [5] or Wein [17]). This sequencing rule will allow the factory to produce more jobs per unit time and thus would eventually provide more timely customer service than a myopic sequencing rule that is based on due-dates. (Notice that the above argument *does not hold* for a factory with only a single machine. This is because, in a single-server queueing system, every work-conserving sequencing rule achieves the same server utilization.

In summary, the research undertaken here attempts to realistically incorporate the dynamic and stochastic elements that are inherent in all factory scheduling problems. Furthermore, we believe that factories, by focusing on system performance measures (such as WIP inventory and cycle time) rather than due-date performance measures, can take advantage of some benefits of Just-In-Time manufacturing and can provide better customer service over the long run.

2. The Limiting Control Problem

We assume readers are familiar with the approximating Brownian network model put forth in Harrison [4]; Most of that paper's notation will be retained for ease of reference. It follows from Section 9 of Harrison [4] that under the balanced heavy loading assumptions, the queueing network scheduling problem described in the introduction can be approximated by the following *limiting control problem*: choose a pair of RCLL processes Y and θ (K-dimensional and one-dimensional, respectively) to

$$\text{minimize} \quad \limsup_{T\to\infty} \frac{1}{T} E_x[\int_0^T \sum_{k=1}^K c_k Z_k(t)dt] \tag{2.1}$$

$$\text{subject to} \quad Y \text{ and } \theta \text{ are non} - \text{anticipating with respect to } X, \tag{2.2}$$

$$Z(t) = X(t) + RY(t) - q\theta(t) \quad \text{for all } t \geq 0, \tag{2.3}$$

$$U(t) = AY(t) \quad \text{for all } t \geq 0, \tag{2.4}$$

$$U \text{ is non} - \text{decreasing with } U(0) = 0, \tag{2.5}$$

$$Z(t) \geq 0 \quad \text{for all } t \geq 0, \text{ and} \tag{2.6}$$

$$\limsup_{T\to\infty} \frac{1}{T} E[U_i(T)] \leq \gamma_i \quad \text{for } i = 1, 2. \tag{2.7}$$

The process Z represents the K-dimensional scaled *queue length* process and describes the state of the system. The K-dimensional process Y represents the scaled centered *allocation* process and the one-dimensional process θ represents the scaled centered *input* process. These two control processes correspond to the sequencing and input decisions, respectively. Interested readers are referred to Harrison [4] for an explicit definition of the process Y, since the definition will not be needed here. As in Harrison [4], exactly the same notation used for the scaled processes are used in defining the approximating Brownian control problem. This is done in order to emphasize the queueing network interpretation

of the Brownian network model. The scaled process θ is defined by

$$\theta(t) = \frac{\bar{\lambda}nt - N(nt)}{\sqrt{n}}, \ t \geq 0, \tag{2.8}$$

where, as mentioned earlier, $\bar{\lambda}$ is the specified average throughput rate, $N(t)$ is the cumulative number of customers released into the network in $[0, t]$ and n is the large integer specified in the balanced heavy loading condition.

The two-dimensional process U represents the scaled *cumulative idleness* process for the two stations. (For brevity's sake, processes such as Z, Y, $U$and θ will often be referred to without the adjective "scaled".) The $K \times K$ *input-output* matrix $R = (R_{kj})$ is defined by

$$R_{kj} = m_j^{-1}(\delta_{jk} - P_{jk}), \tag{2.9}$$

where δ_{jk} denotes the Dirac delta function, meaning that $\delta_{jk} = 1$ if $j = k$ and $\delta_{jk} = 0$ otherwise. The $2 \times K$ *resource consumption* matrix $A = (A_{ik})$ is defined by

$$A_{ik} = \begin{cases} 1, & \text{if i = s(k);} \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

The K-dimensional process X is a (δ, Σ) Brownian motion, but several definitions are needed before stating the K-dimensional drift vector $\delta = (\delta_k)$ and the $K \times K$ covariance matrix $\Sigma = (\Sigma_{jl})$. Let $\lambda = (\lambda_k)$ be defined by

$$\lambda = q\bar{\lambda}, \tag{2.11}$$

so that λ_k represents the average number of class k customers that must arrive to the system per unit of time in order to satisfy the throughput rate constraint.

Since P was assumed to be transient, it follows that R is non-singular and there exists a unique non-negative $K-$vector $\beta = (\beta_k)$ satisfying the flow balance equations

$$\lambda = R\beta. \tag{2.12}$$

Letting $C(i)$ be the set of all customer classes k such that $s(k) = i$, define the two-vector of traffic intensities $\rho = (\rho_i)$ by

$$\rho_i = \sum_{k \in C(i)} \beta_k. \tag{2.13}$$

Now define the K-vector $\alpha = (\alpha_k)$ by

$$\alpha_k = \frac{\beta_k}{\rho_i} \text{ for all } k \in C(i). \tag{2.14}$$

Then the drift δ and covariance Σ of the Brownian motion X are

$$\delta = +\sqrt{n}(\lambda - R\alpha) \text{ and} \tag{2.15}$$

$$\Sigma_{jl} = \sum_{k=1}^{K} [\alpha_k m_k^{-1} P_{kj}(\delta_{jl} - P_{kl}) + \alpha_k m_k^{-1} s_k^2 R_{jk} R_{lk}]. \tag{2.16}$$

Inequality (2.7), which expresses the throughput rate constraint in terms of the cumulative server idleness process U, is the only relationship in the limiting control problem that does not appear in the Brownian network formulation of [4]. The two-vector $\gamma = (\gamma_i)$ in (2.7) is defined by

$$\gamma_i = \sqrt{n}(1 - \rho_i). \tag{2.17}$$

To derive (2.7), let the $2 \times K$ matrix $M = (M_{ik})$ be defined by

$$M = AR^{-1}. \tag{2.18}$$

M is called the *workload profile matrix*, and M_{ik} is interpreted as the expected total amount of time that server i must devote to a class k customer before that customer exits the network. Define the two-dimensional vector $v = (v_i)$ by

$$v = Mq, \tag{2.19}$$

so that v_i is interpreted as the expected total amount of time over the long-run that server i spends on each customer. From (2.11)-(2.13) and (2.18)-(2.19), it follows that

$$\rho_i = v_i \bar{\lambda} \text{ for } i = 1, 2. \tag{2.20}$$

Inequality (2.7) follows from (2.17) and (2.20), since the long-run average throughput rate is greater than or equal to $\bar{\lambda}$ if and only if the long-run average fraction of time that server i is idle is less than or equal to $1 - \rho_i$.

3. The Workload Formulation

The state of the system in the limiting control problem is described by a $K-$dimensional queue length process, by way of the basic system relationship (2.3). In this section the limiting control problem is reformulated so that the state of the system is described by a two-dimensional workload process. Recalling the definition (2.18) of the workload profile matrix M, let us define the two-dimensional scaled *workload process* $W = (W_i)$ by

$$W(t) = MZ(t), \ t \geq 0, \tag{3.1}$$

where $W_i(t)$ is interpreted as the expected total amount of work for server i embodied in those customers who are present anywhere in the network at time t. Define the two-dimensional Brownian motion $B(t) = (B_i(t))$ by

$$B(t) = MX(t), \ t \geq 0. \tag{3.2}$$

The process B has drift $M\delta$ and covariance $M\Sigma M^T$. By (2.10), (2.12)-(2.15) and (2.17)-(2.18), one can show that the two-dimensional drift vector $M\delta = -\gamma$.

Define the *workload formulation* of the limiting control problem as choosing RCLL processes Z, U and θ (K-, two- and one-dimensional, respectively) so as to

$$\text{minimize} \quad \limsup_{T\to\infty} \frac{1}{T} E_x[\int_0^T \sum_{k=1}^K c_k Z_k(t)dt] \tag{3.3}$$

$$\text{subject to} \quad U \ \text{ and } \theta \text{ are non} - \text{anticipating with respect to } B, \tag{3.4}$$

$$U \text{ is non} - \text{decreasing with } U(0) = 0, \tag{3.5}$$

$$Z(t) \geq 0 \quad \text{for all } t \geq 0, \tag{3.6}$$

$$\limsup_{T\to\infty} \frac{1}{T} E[U_i(T)] \leq \gamma_i \quad \text{for } i = 1, 2, \quad \text{and} \tag{3.7}$$

$$MZ(t) = B(t) + U(t) - v\theta(t) \quad \text{for all} \quad t \geq 0. \tag{3.8}$$

Let us call a pair of RCLL processes (Y, θ) a *feasible policy* for the limiting control problem if it satisfies equations (2.3)-(2.7) and call a triple of RCLL processes (Z, U, θ) a feasible policy for the workload formulation if it satisfies equations (3.5)-(3.8). The following proposition, which was proved in Wein [17], allows us to analyze the workload formulation of the limiting control problem, rather than studying problem (2.1)-(2.7) directly.

Proposition 2.1. *Every feasible policy (Y, θ) for the limiting control problem yields a corresponding feasible policy (Z, U, θ) for the workload formulation and every feasible policy (Z, U, θ) yields a corresponding feasible policy (Y, θ).*

It was shown in Wein [17] that if the control process Y is non-anticipating with respect to the Brownian motion X in the limiting control problem, then the control process U is non-anticipating with respect to the Brownian motion B in the workload formulation. It was also shown that the solution to the workload formulation remains unchanged whether θ is non-anticipating with respect to X or with respect to B.

4. Solution to the Workload Formulation

The solution (U, Z, θ) to the workload formulation (3.3)-(3.8) of the limiting control problem was derived in Wein [17]. This is a self-contained section that summarizes the solution. The parameters ρ_i, M_{ik} and v_1 appearing in this section are all defined in terms of the primitive problem data by definitions (2.13), (2.18) and (2.19), respectively. We also need to define the parameters σ^2, h_1, h_2, ν and ξ, which can be calculated in terms of the

primitive problem data. The parameter σ^2 is defined by $\sigma^2 = \varrho^T M \Sigma M^T \varrho$, where

$$\varrho = \begin{bmatrix} \rho_2 \\ -\rho_1 \end{bmatrix}. \tag{4.1}$$

Without loss of generality, assume that the classes $k = 1, ..., K$ are ordered so that

$$\arg\max_k \; c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}) = 1 \tag{4.2}$$

and

$$\arg\min_k \; c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}) = 2. \tag{4.3}$$

Now define the positive coefficients h_1 and h_2 by

$$h_1 = \frac{c_2}{\rho_1 M_{22} - \rho_2 M_{12}} \tag{4.4}$$

and

$$h_2 = \frac{c_1}{\rho_2 M_{11} - \rho_1 M_{21}}. \tag{4.5}$$

Finally, let

$$\nu = \frac{2sqrtn(\rho_1 - \rho_2)}{\sigma^2} \tag{4.6}$$

and

$$\xi = \sqrt{n}\rho_1(1 - \rho_1). \tag{4.7}$$

In the workload formulation, the controller observes a two-dimensional Brownian motion process B, from which can be observed the one-dimensional Brownian motion process $\hat{B}$ defined by

$$\hat{B}(t) = \rho_2 B_1(t) - \rho_1 B_2(t), \; t \geq 0. \tag{4.8}$$

If $\rho_1 \neq \rho_2$, then define the interval endpoints a and b by

$$a = \nu^{-1} \ln\left(\frac{(h_1 + h_2)\rho_2(1 - \rho_1)}{h_1\rho_2(1 - \rho_1) + h_2\rho_1(1 - \rho_2)}\right) \tag{4.9}$$

and

$$b = \nu^{-1} \ln\left(\frac{(h_1 + h_2)\rho_1(1 - \rho_2)}{h_1\rho_2(1 - \rho_1) + h_2\rho_1(1 - \rho_2)}\right). \tag{4.10}$$

If $\rho_1 = \rho_2$, then let

$$a = -\frac{h_2}{h_1 + h_2}\frac{\sigma^2}{2\xi} \tag{4.11}$$

and

$$b = \frac{h_1}{h_1 + h_2}\frac{\sigma^2}{2\xi}. \tag{4.12}$$

For a particular realization of $\hat{B}$, define the control functionals (R, L) by

$$R(t) = \sup_{0 \le s \le t} [a - \hat{B}(s) + L(s)]^+ \tag{4.13}$$

and

$$L(t) = \sup_{0 \le s \le t} [\hat{B}(s) + R(s) - b]^+. \tag{4.14}$$

The two-dimensional optimal control process U is given by

$$U_1(t) = \frac{R(t)}{\rho_2} \tag{4.15}$$

and

$$U_2(t) = \frac{L(t)}{\rho_1}. \tag{4.16}$$

From the functionals (R, L) in (4.13)-(4.14), next define the process $\hat{W}$ by

$$\hat{W}(t) = \hat{B}(t) + R(t) - L(t) \quad \text{for all } t \ge 0. \tag{4.17}$$

The K-dimensional optimal control process Z is given by

$$Z_k(t) = \begin{cases} \frac{\hat{W}(t)}{\rho_2 M_{11} - \rho_1 M_{21}}, & \text{if } k = 1 \text{ and } \hat{W}(t) \ge 0; \\ 0, & \text{if } k \ne 1 \text{ and } \hat{W}(t) \ge 0. \end{cases} \tag{4.18}$$

and

$$Z_k(t) = \begin{cases} \frac{\hat{W}(t)}{\rho_2 M_{12} - \rho_1 M_{22}}, & \text{if } k = 2 \text{ and } \hat{W}(t) < 0; \\ 0, & \text{if } k \ne 2 \text{ and } \hat{W}(t) < 0. \end{cases} \tag{4.19}$$

Finally, the optimal control process θ is given by

$$\theta(t) = v_1^{-1}[B_1(t) + \frac{R(t)}{\rho_2} - \sum_{k=1}^{K} M_{1k} Z_k(t)], \quad \text{for all } t \ge 0. \tag{4.20}$$

Thus the solution (U, Z, θ) to the workload formulation (3.3)-(3.8) is given by equations (4.15)-(4.16) and(4.18)-(4.20). In the next two sections, this solution will be interpreted in terms of the queueing network model.

5. The Sequencing Rule

In this section we describe the sequencing rule, which is based on the control process Z. Consider again the workload formulation (3.3)-(3.8) of the limiting control problem. According to definition (3.1), we must have $W(t) = MZ(t)$. This means that at any time t, the scaled queue length process Z can be any nonnegative vector that is consistent with the present scaled workload process W. Thus, in the idealized Brownian approximation, queues of different customer classes can be instantaneously swapped for one another, as long as the expected work content remains unchanged. These swaps, which can be interpreted as the reallocation of server time among the various classes, appear to occur instantaneously because we are observing the system evolving in *scaled* time.

From the solution Z in (4.18)-(4.19), it is seen that only two of the K components of Z are ever positive. These two components correspond to the two customer classes that are

$$\arg\max_k \; c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}) \tag{5.1}$$

and

$$\arg\min_k \; c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}), \tag{5.2}$$

which were denoted by classes 1 and 2, respectively, by conventions (4.2)-(4.3). Furthermore, at each time t, only one customer class has a positive queue length. According to formulas (4.18)-(4.19), class 1 customers have a positive queue length whenever the workload imbalance $\hat{W}(t) > 0$, and class 2 customers have a positive queue length whenever $\hat{W}(t) < 0$. In the case where $c_k = c$ for all $k = 1, ..., K$ (i.e., the objective is to minimize

the long-run average cycle time of customers), it is true that class 1 is served at station 1 and class 2 is served at station 2. This is interpreted to mean that whenever $\hat{W}(t) > 0$, customers of class 1 are only served when there are no other customers present at station 1. Similarly, whenever $\hat{W}(t) < 0$, customers of class 2 are only served when there are no other customers present at station 2.

Under heavy traffic conditions, it does not matter in what order classes $2, ..., K$ are served when $\hat{W}(t) > 0$, or in what order classes $1, 3, ..., K$ are served when $\hat{W}(t) < 0$; it is only required that the two servers be kept busy when there is work for them to do. There are two reasons for this. The first reason, as will be seen in the next section, is that the asymptotically optimal input rule prevents a large queue of customers from forming at station 1 (respectively, station 2) when $\hat{W}(t) < 0$ (respectively, $\hat{W}(t) > 0$). Consequently, in the scaled space of the Brownian limit, all customers at station 1 (respectively, station 2) vanish when $\hat{W}(t) < 0$ (respectively, $\hat{W}(t) > 0$). The second reason is because the customer classes that are not given bottom priority will not see the queueing system in a heavy traffic situation, and thus their scaled queue lengths will be negligible compared to that of the bottom priority classes. This phenomenon of the normalized queue length processes of high priority customers vanishing in the heavy traffic limit has been observed in previous work. Whitt [18], Harrison [3], and Reiman [12] have obtained heavy traffic limit theorems in a single station system, and Johnson [7] and Peterson [11] have obtained similar results in a network setting. However, a formal limit theorem has yet to be proved for our case of a general multiclass network with feedback.

To repeat, the interpretation of formulas (4.18)-(4.19) is to give class 1 customers lowest priority at station 1 when $\hat{W}(t) > 0$ and give class 2 customers lowest priority at station 2 when $\hat{W}(t) < 0$. There seems to be some ambiguity that remains in specifying a sequencing rule that emerges from the solution of the Brownian control problem. However, from (4.1)-(4.2), when $c_k = c$ for all $k = 1, ..., K$, there is a natural ranking of the K customer classes by the index $\rho_2 M_{1k} - \rho_1 M_{2k}$. We now propose two sequencing policies

that give class 1 (respectively, class 2) customers lowest priority at station 1 (respectively, station 2) when $\hat{W}(t) > 0$ (respectively, $\hat{W}(t) < 0$). The first policy is a static priority rule that awards higher priority at station 1 (respectively, station 2) to the classes with the smaller (respectively, larger) values of the index $\rho_2 M_{1k} - \rho_1 M_{2k}$.

The second policy is obtained by computing *dynamic reduced costs* for each customer class $k = 1, ..., K$. The reduced cost for a class k customer at time t can be interpreted as the increase in the objective function of the linear program (originally stated as equations (3.1)-(3.4) in Wein [17])

$$\min_{Z(t),\theta(t)} \quad \sum_{k=1}^{K} c_k Z_k(t) \tag{5.3}$$

$$\text{subject to} \quad \sum_{k=1}^{K} M_{1k} Z_k(t) + v_1 \theta(t) = B_1(t) + U_1(t) \tag{5.4}$$

$$\sum_{k=1}^{K} M_{2k} Z_k(t) + v_2 \theta(t) = B_2(t) + U_2(t) \tag{5.5}$$

$$Z_k(t) \geq 0, \quad \text{for } k = 1, ..., K \tag{5.6}$$

per unit increase in the righthand side of the nonnegativity constraint $Z_k(t) \geq 0$. It was shown in Wein [17] that the partial solution $Z(t)$ to this linear program yields the optimal control process Z given in equations (4.18)-(4.19). It was also shown there that the dual of (5.3)-(5.6) can be expressed as

$$\max_{\pi_1(t)} \quad \frac{\hat{W}(t)}{\rho_2} \pi_1(t) \tag{5.7}$$

$$\text{subject to} \quad c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}) \pi_1(t) \leq \rho_2 \quad \text{for } k = 1, ..., K. \tag{5.8}$$

The reduced costs at time t for the K variables in the dual of (5.7)-(5.8) are

$$\rho_2 - c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k}) \pi_1^*(t) \quad \text{for } k = 1, ..., K, \tag{5.9}$$

where $\pi_1^*(t)$ is the solution to (5.7)-(5.8). The higher the value of the k-th reduced cost in (5.9), the more expensive it is to hold class k customers in the queue. Furthermore, the

reduced cost for a class k customer is zero when $Z_k(t) > 0$ in (4.18)-(4.19). Let us again assume that $c_k = c$ for $k = 1, ..., K$ and consider the policy that gives *highest* priority at each time t to the customer class with the *largest* reduced cost. Then one obtains a dynamic scheduling rule that ranks all K classes by the index $\rho_2 M_{1k} - \rho_1 M_{2k}$ and, at each station, serves the class with the smallest (respectively, largest) value of the index when $\hat{W}(t) > 0$ (respectively, $\hat{W}(t) < 0$).

Simulation results (see Section 7) on several systems have indicated that both the static priority rule and the dynamic rule work well in conjunction with the workload regulating input rule described in the next section. The static rule has the advantage that it is easier to implement, since it does not depend on any global state information. The dynamic rule has the advantage that it has a natural generalization to networks with more than two bottleneck stations.

Thus far in this section it has been assumed that $c_k = c$ for all $k = 1, ..., K$. If we relax this assumption, it does not necessarily follow that class 1 is served at station 1 and class 2 is served at station 2. When this is indeed still the case, then the same two priority rules described earlier, now based on the index $c_k^{-1}(\rho_2 M_{1k} - \rho_1 M_{2k})$, are the proposed sequencing policies. If this is not the case, the equations (4.18)-(4.19) still suggest giving class 1 customers lowest priority when $\hat{W}(t) > 0$ and giving class 2 customers lowest priority when $\hat{W}(t) < 0$. However, additional simulation studies need to be performed before making a more specific policy recommendation.

6. The Input Rule

In this section the input rule, which is based on the control processes U and θ, is described. In the workload formulation of the limiting control problem, the controller observes the two-dimensional Brownian motion process B, exerts the controls U and θ,

and obtains the controlled process W, which is the scaled workload process. The basic system state equations (3.8) that govern the controlled process can be expressed as

$$W_1(t) = B_1(t) + U_1(t) - v_1\theta(t) \quad \text{and} \tag{6.1}$$

$$W_2(t) = B_2(t) + U_2(t) - v_2\theta(t). \tag{6.2}$$

Since these equations are linear and additive, the controls U and θ act as "pushes" on the Brownian motion B. Recall that the non-decreasing process U_i represents the scaled cumulative idleness process for station i. Also, θ is the scaled centered input process, and the vector v is proportional to the server utilization levels ρ.

Since $W = MZ$, the solution Z in (4.18)-(4.19) implies that the workload process W resides on the boundary of a cone in the nonnegative orthant of $\mathbf{R^2}$. From equations (4.18)-(4.19), it can be seen that the control U_1 (respectively, U_2) is exerted only when the scaled workload imbalance process $\hat{W}$ equals a (respectively, b). Exerting the control U_i is interpreted as incurring server idleness at station i.

In terms of the two-dimensional workload process W, the interval endpoints a and b correspond to reflecting barriers on the boundary of the cone, beyond which W may not enter. This situation is depicted in Figure 1, where W must reside on the portion of the cone boundary that is in boldface. In the optimal solution, the controls U_1 and U_2 are only exerted when $W_2(t) = c_2^*$ and $W_1(t) = c_1^*$, respectively, where the scaled *threshold levels* c_1^* and c_2^* can be calculated explicitly from the solution to the workload formulation. Otherwise, only the input process θ is used to keep the controlled process W on the boundary of the cone. Thus, the policy that emerges from the Brownian control problem attempts to manipulate input *in lieu of idling servers* and keeps the workload process W on the boldface portion of the cone boundary in Figure 1. However, when the process W reaches the barrier at c_1^* or the barrier at c_2^* in Figure 1, then the controller refuses to release any more customers into the system and is willing to incur server idleness.

In order to see exactly how the input is manipulated, recall that by equation (2.19)

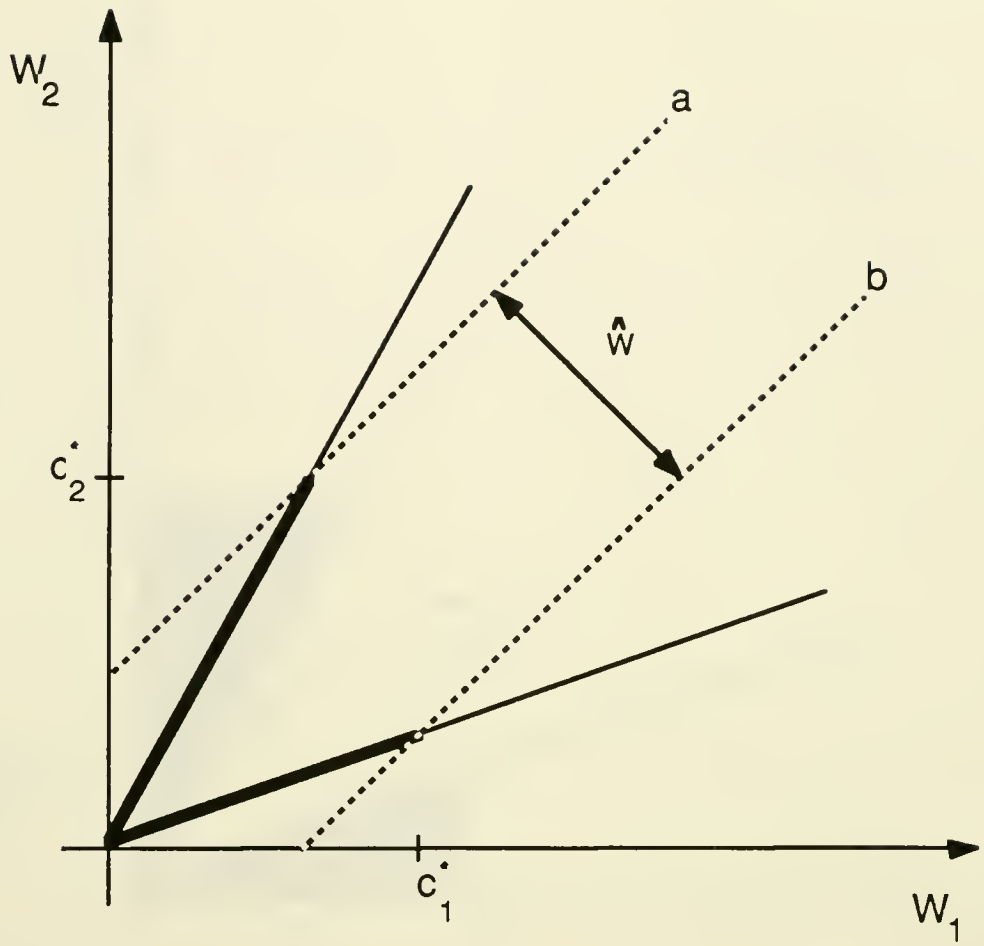

FIGURE 1

and the balanced loading conditions, v_1 is approximately equal to v_2 and so the scaled centered input process θ can move along a direction that is close to the 45 degree line. The process θ was defined in equation (2.8) by

$$\theta(t) = \frac{\bar{\lambda} nt - N(nt)}{\sqrt{n}}, \; t \geq 0, \tag{6.3}$$

where the process N is the cumulative number of customers released into the system up to time t. Thus, when θ moves in the negative 45 degree direction, input is being witheld *relative to the nominal* input rate, and when θ moves in the positive 45 degree direction, input is being increased relative to the nominal input rate. This is depicted in Figure 2, where input is witheld whenever the workload process is in the cone and input is increased

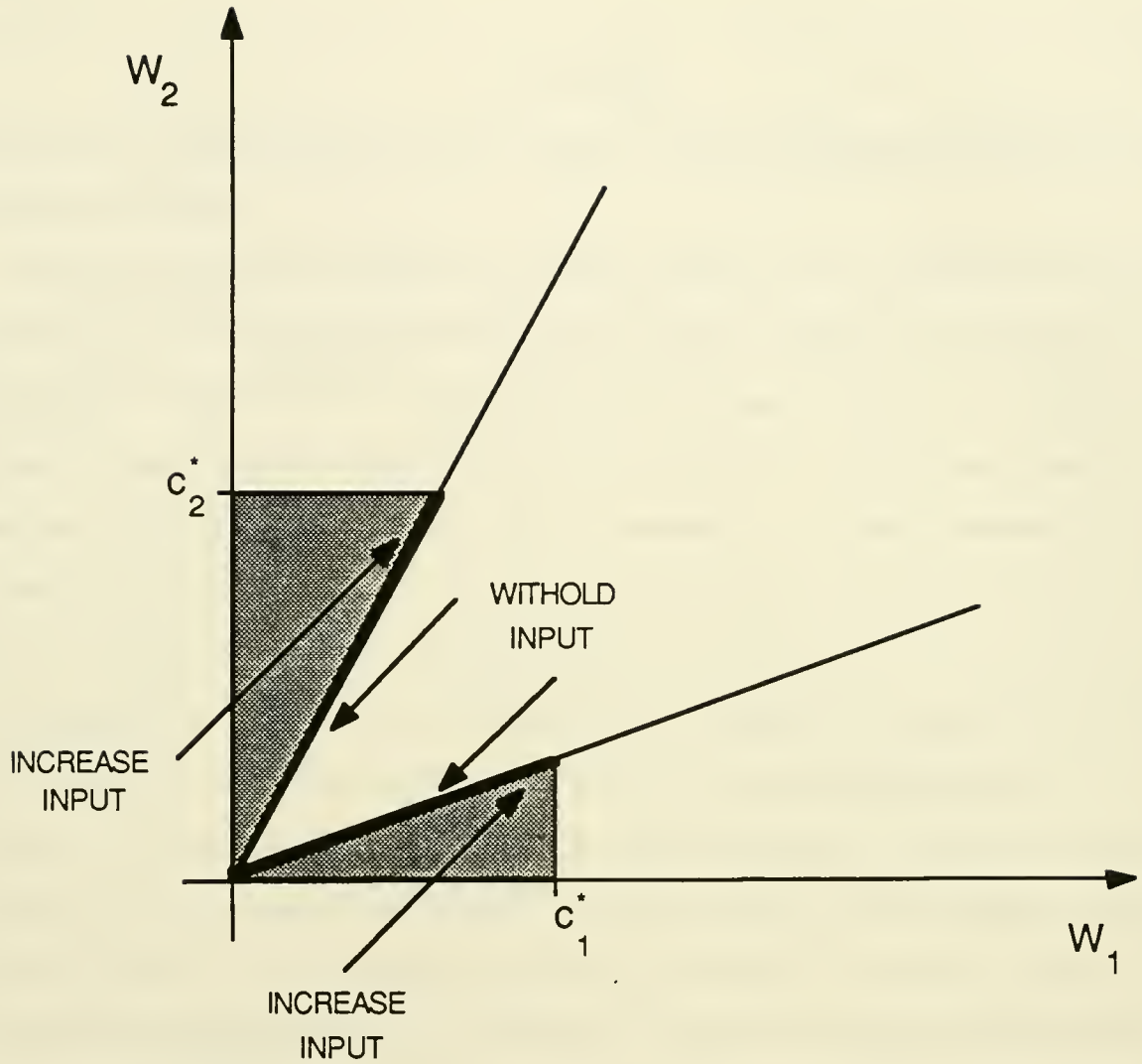

FIGURE 2

whenever the workload process is in the shaded region.

Notice that in the actual queueing system, it may be possible for the workload process to reside outside of the cone. This is because the state space of W is the cone $\{W = MZ, Z \geq 0\}$, which contains the cone pictured in Figure 1. Its extremal rays are generated by the two customer classes

$$\arg\max_k \frac{M_{1k}}{M_{2k}} \tag{6.4}$$

and

$$\arg\min_k \frac{M_{1k}}{M_{2k}}, \tag{6.5}$$

which may not coincide with the rays in Figure 1, which are generated by the two classes defined in (5.1)-(5.2).

The main goal of this section is to develop an effective input policy for the actual queueing system that operationalizes the optimal solution obtained from the limiting control problem. To this end, let us interpret the word "increase" in Figure 2 to simply mean "release a customer into the system" and the word "withold" to simply mean "cease input". Then the naive rule that emerges from this interpretation is to release a customer into the system when the workload process W enters the shaded region in Figure 2. However, this naive rule ignores a major difference that exists between the actual queueing system and the idealized heavy traffic limit. This difference can be understood by making the following observation. In the idealized Brownian setting, when the scaled workload process W is on the lower ray of the cone boundary and $W_1(t) < c_1^*$, then there are zero scaled customers at station 2 and yet *station 2 is not idle.* Similarly, when W is on the upper ray of the cone and $W_2(t) < c_2^*$, then there are zero customers at station 1 and station 1 is not idle. This apparent paradox is due to the rescaling that occurs when passing to the heavy traffic limit. In the actual queueing system, there are enough customers at the particular station to avoid idleness, but when looked at in the scaled space of the heavy traffic limit, these customers vanish.

In order to adapt the naive control rule stated above to the actual queueing system, it is necessary to build in a boundary layer of thickness ϵ on the inside of the cone boundary, as shown in Figure 3. This boundary layer generates a new cone, which we call the ϵ-*cone*, that is strictly within the original cone. The input rule is still to release a customer into the system whenever the workload process enters the shaded region, but now the shaded region is enlarged by including the area between the two cones, as in Figure 3. This layer, which is negligible in scaled space, prevents the process W from straying very far from the

boldfaced portion of the original cone boundary, but allows the servers to be utilized the requisite portion of the time. As ϵ increases, the servers will incur less idleness but the queue lengths may grow as a result. In an actual queueing system, the appropriate setting of ϵ will depend on the amount of variability in the queueing system and the amount of time customers spend at non-bottleneck stations. In fact, one could use a layer of thickness ϵ_1 on the lower ray of the cone boundary, and a layer of thickness ϵ_2 on the upper ray of the cone boundary.

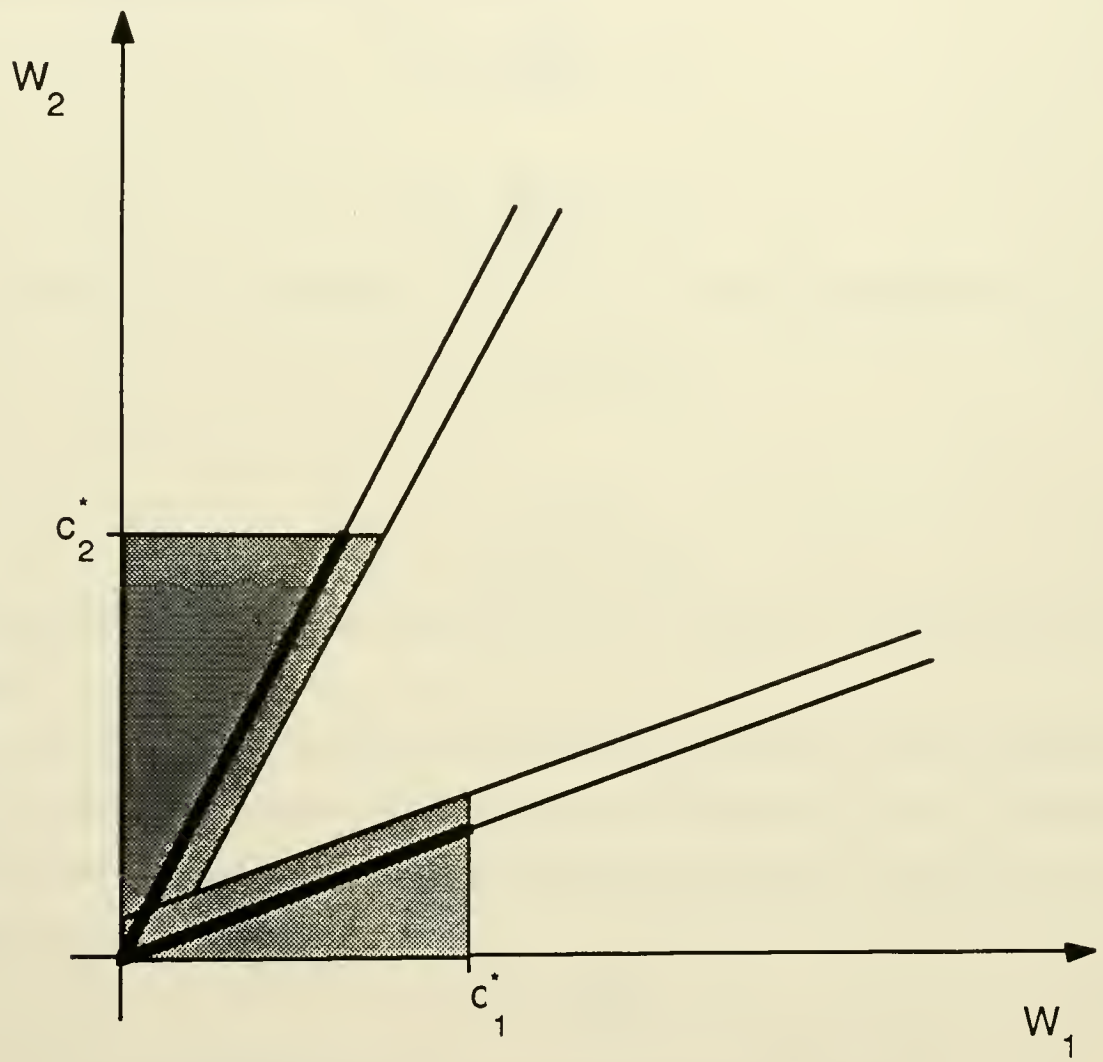

FIGURE 3

Thus the suggested input rule is to release a customer whenever the workload process enters the shaded region of Figure 3. This region can be calculated explicitly in terms of the problem data and the parameters ϵ_1 and ϵ_2. The cone in Figure 1 is generated by the rays

$$W_2 - \frac{M_{21}}{M_{11}} W_1 = 0 \tag{6.6}$$

and

$$W_1 - \frac{M_{12}}{M_{22}} W_2 = 0. \tag{6.7}$$

Therefore the regions outside of the ϵ-cone are

$$W_2 - \frac{M_{21}}{M_{11}} W_1 \leq \epsilon_1 \tag{6.8}$$

and

$$W_1 - \frac{M_{12}}{M_{22}} W_2 \leq \epsilon_2. \tag{6.9}$$

From (2.37), (3.8) and (4.18)-(4.19), c_1^* and c_2^* can be solved for explicitly. The solution is

$$c_1^* = \frac{M_{11} b}{\rho_2 M_{11} - \rho_1 M_{21}} \tag{6.10}$$

and

$$c_2^* = \frac{M_{22} a}{\rho_2 M_{12} - \rho_1 M_{22}}, \tag{6.11}$$

where a and b are the optimal interval endpoints from the solution to the Brownian control problem.

Notice that W, c_1^* and c_2^* are all in *scaled terms*, and in order to find an appropriate policy for the original queueing system, some *unscaling* needs to be done. By definitions (3.1) and the standard heavy traffic scaling described in Section 5 of Harrison [4], it can be seen that

$$W(t) = \frac{w(nt)}{\sqrt{n}}, \tag{6.12}$$

where w is the *unscaled* workload process defined in the introduction by

$$w(t) = MQ(t), \ t \geq 0, \tag{6.13}$$

and Q is the actual queue length process. Define $w_i^* = \sqrt{n}c_i^*$ for $i = 1, 2$ to be the threshold levels for the input policy. Then the suggested input rule is to release a customer into the system at times t such that either

$$w_1(t) < w_1^* \text{ and} \tag{6.14}$$

$$w_2(t) - \frac{M_{21}}{M_{11}} w_1(t) \leq \epsilon_1, \tag{6.15}$$

or

$$w_2(t) < w_2^* \text{ and} \tag{6.16}$$

$$w_1(t) - \frac{M_{12}}{M_{22}} w_2(t) \leq \epsilon_2. \tag{6.17}$$

Here, ϵ_1 and ϵ_2 are parameters that can be set in order to achieve a desired output rate. As will be seen in the next section, the setting of these parameters is quite simple, at least when there are no non-bottleneck stations in the queueing system.

7. An Example

The scheduling rules stated in Sections 5 and 6 will be illustrated by means of an example. The example will have two customer types, A and B, and there is a 50-50 product mix that is specified, so that customers are released into the system in the order ABABAB... As seen in Figure 4, customer type A has two stages on its route and customer type B has four stages. The six customer classes are designated (and ordered from $k = 1, ..., 6$) by A1, A2, B1, B2, B3 and B4, since each class corresponds to a type-stage pair.

The mean service times (in arbitrary time units) for each customer class are indicated in Figure 4. For concreteness (since simulation results will be exhibited), all service times are assumed to be exponential, although our results hold for any service time distributions with finite mean and variance. Calculation of the 2×6 workload profile matrix M yields

$$M = \begin{pmatrix} 4 & 0 & 10 & 2 & 2 & 0 \\ 1 & 1 & 13 & 13 & 7 & 7 \end{pmatrix}. \tag{7.1}$$

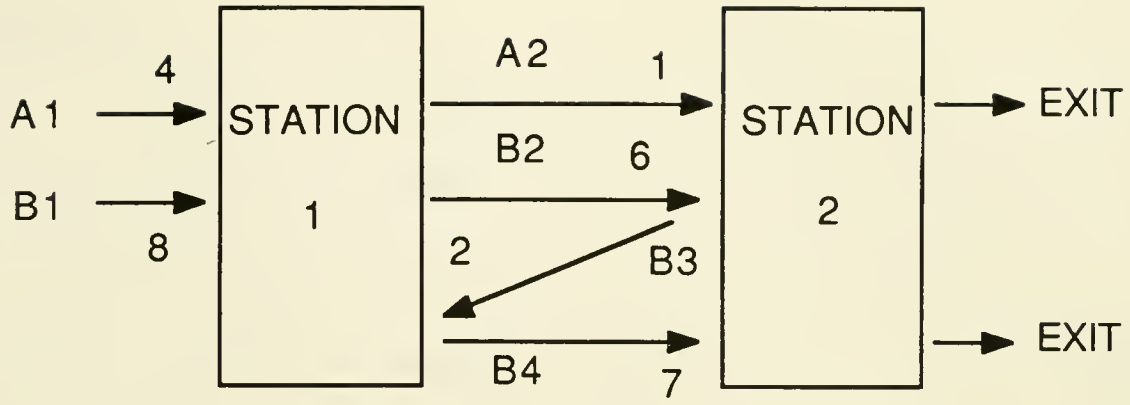

FIGURE 4

From (2.19), $v_1 = v_2 = 7$, so assuming $\rho_1 = \rho_2 = .9$ (and letting $n = 100$), we obtain a target long-run average output rate $\bar{\lambda} = .1286$ customers per unit time. As in the original problem formulation, the objective is to minimize the long-run average cycle time of customers subject to meeting the average output rate of .1286 customers per unit time. From (7.1), one obtains the indices (since $\rho_1 = \rho_2$, they need not enter into the indices)

$$M_{1k} - M_{2k} = (3 \ -1 \ -3 \ -11 \ -5 \ -7) \qquad \text{for } k = 1, ..., 6. \tag{7.2}$$

Thus, the suggested static sequencing rule gives priorities (from highest to lowest) in the order (B3, B1, A1) at station 1 and (A2, B4, B2) at station 2.

In order to find the suggested input policy, the interval endpoints a and b need to be found. Using formulas (4.11)-(4.12) yields $a = -.4365\sigma^2$ and $b = .119\sigma^2$. Going through the necessary calculations in (2.16), (3.2) and (4.1) gives $\sigma^2 = 10.93$. Using equations (6.10)-(6.11), one can compute the values of $c_1^* = 1.93$ and $c_2^* = 6.26$. Upon unscaling, the threshold levels are found to be $w_1^* = 19.3$ time units of work and $w_2^* = 62.6$ time units of work.

Since W only takes on integer values in our example, it follows from equations (6.14)-(6.17) that the suggested input rule is to release a customer into the system at times t

such that either

$$w_1(t) \leq 19 \quad \text{and} \tag{7.3}$$

$$w_2(t) - \frac{1}{4} w_1(t) \leq \epsilon_1, \tag{7.4}$$

or

$$w_2(t) \leq 62 \quad \text{and} \tag{7.5}$$

$$w_1(t) - \frac{2}{13} w_2(t) \leq \epsilon_2. \tag{7.6}$$

Using the example network, a simulation study was undertaken to compare the performance of the suggested scheduling rule against conventional input and sequencing rules. Three input rules were tested: the suggested input rule (abbreviated by $WR(\epsilon_1, \epsilon_2)$ for workload regulating input, where ϵ_1 and ϵ_2 are the boundary layer thicknesses used); closed loop input (abbreviated by $CL(N)$, where N is the total number of customers in the network); and deterministic input, where the interarrival times are constant. For all input rules, customers entered the system in the order ABABAB... Five sequencing rules were compared: first-in first-out (FIFO); shortest expected processing time (SPT); shortest expected remaining processing time (SRPT); the asymptotically optimal sequencing rule (abbreviated by ST($M_1 - M_2$)); and the rule based on the dynamic reduced costs that was described in Section 5 (abbreviated by DY($M_1 - M_2$)). Another common rule in the scheduling literature is the least work next queue (LWNQ) rule. This policy gives priority to the customer who is going next to the queue that has the least expected amount of work in it. The LWNQ rule is not relevant here, since all customers at station 1 go next to station 2, and all customers at station 2 go next to station 1 or exit the system.

The results of the simulation study are summarized in Table 1. Each row gives statistics for a particular scheduling policy, which is specified by a particular input control rule paired with a specific sequencing rule. The first two columns of Table 1 state the scheduling policy. For each policy tested, ten independent runs were made, each consisting of 2000

customer completions. The third column gives the average throughput rate (in customers per unit time) over the ten runs, along with a 95% confidence interval. The fourth column of Table 1 contains the average cycle time of customers over the ten runs, along with a 95% confidence interval for this value. Rather than use the target average throughput rate $\bar{\lambda} = .1286$ customers per unit time, it was more convenient to choose the parameters of the closed input policies and workload regulating input policies so as to achieve a throughput rate of .127 customers per unit time. This average throughput rate corresponds to an average server utilization of 88.9%. To allow for easy comparisons of the average cycle times for the various policies, all simulation runs achieved this target output rate.

Each simulation run had no initialization period, and all runs began with an empty system. For closed loop input runs, the customers arrived according to deterministic input (at the same rate as the corresponding open models) until the network had reached its population limit, and then closed loop input was used.

Referring to the results in Table 1, it is seen that workload regulating input in combination with either of the sequencing rules described in Section 5 easily outperformed all other combinations of input and sequencing rules. The difference in performance between the ST($M_1 - M_2$) and DY($M_1 - M_2$) rules was not statistically significant. They both achieved nearly a 30% reduction in average cycle time, compared to the next best scheduling rule, which was the closed loop input in combination with the $ST(M_1 - M_2)$ sequencing rule. This sequencing rule, which was shown in Harrison and Wein [5] to maximize the throughput rate of a two-station closed queueing network in heavy traffic, achieved a 30% reduction in average cycle time compared to FIFO in the closed loop input case.

Since the workload regulating input rule was derived jointly with the $ST(M_1 - M_2)$ and $DY(M_1 - M_2)$ rules, the input rule was not tested in combination with the other three sequencing rules. Similarly, the $ST(M_1 - M_2)$ and $DY(M_1 - M_2)$ rules were not tested in combination with input rules with which they were not derived.

INPUT RULE	SEQUENCING RULE	THROUGHPUT RATE (95% C.I.)	CYCLE TIME (95% C.I.)
DETERMINISTIC	FIFO	.127(±.000)	92.0(±4.1)
DETERMINISTIC	SPT	.127(±.000)	73.8(±3.1)
DETERMINISTIC	SRPT	.127(±.000)	66.6(±2.5)
CL(10)	FIFO	.127(±.001)	78.6(±0.7)
CL(10)	SPT	.127(±.001)	78.2(±0.8)
CL(8)	SRPT	.127(±.001)	62.5(±0.6)
CL(7)	ST($M_1 - M_2$)	.127(±.001)	54.9(±0.4)
WR(1,1)	ST($M_1 - M_2$)	.127(±.001)	38.6(±0.9)
WR(1,1)	DY($M_1 - M_2$)	.127(±.001)	38.9(±0.9)

TABLE 1

The ease of implementation and accuracy of the workload regulating input rule was even more impressive than its actual performance. Values of $\epsilon_1 = \epsilon_2 = 1$ achieved the target output rate of .127 customers per unit time. Furthermore, in order to test the accuracy of the threshold levels of 19 and 62 derived in (7.3) and (7.5) from the Brownian control problem, a search over all values of threshold levels was made, while keeping $\epsilon_1 = \epsilon_2 = 1$. The best threshold levels achieved only a 2.1% improvement over the derived values of 19 and 62. Thus, although the Brownian network model appears to be a rather crude model at first glance, its results are surprisingly accurate, at least when there are no non-bottleneck stations present in the network.

Acknowledgements

I am deeply indebted to my dissertation advisor, J. Michael Harrison. He suggested this problem area to me, and provided invaluable suggestions on both the technical and expository aspects of this paper. This research was partially supported by the Semiconductor Research Corporation and by a predoctoral fellowship from the International Business Machines Corporation.

REFERENCES

[1] Chen, H. and Mandelbaum, A. (1987). Stochastic Flow Networks: Bottlenecks and Diffusion Approximations. In preparation.

[2] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967). *Theory of Scheduling.* Addison-Wesley, Reading, Mass.

[3] Harrison, J. M. (1973). A Limit Theorem for Priority Queues in Heavy Traffic. *J. Appl. Prob.* 10, 907-912.

[4] Harrison, J. M. (1987). Brownian Models of Queueing Networks with Heterogeneous Customer Classes. *Proc. IMA Workshop on Stochastic Differential Systems.* Springer-Verlag, Berlin.

[5] Harrison, J. M. and Wein, L. M. (1987). Scheduling a Two-Station Multiclass Closed Queueing Network in Heavy Traffic. Submitted for publication.

[6] Jacobs, R. F. (1983). The OPT Scheduling System: A Review of a New Production Scheduling System. *Production and Inventory Management* 24, 47-51.

[7] Johnson, D. P. (1983). Diffusion Approximations for Optimal Filtering of Jump Processes and for Queueing Networks. Unpublished Ph.D. thesis, Dept. of Mathematics, Univ. of Wisconsin, Madison.

[8] Klimov, G. P. (1974). Time Sharing Service Systems I. *Th. Prob. Appl.* 19,532-551.

[9] Little, J. D. C. (1961). A Proof of the Queueing Formula $L = \lambda W$. *Ops. Rsch.* 9, 383-387.

[10] Morton, T. E. and Smunt, T. L. (1986). A Planning and Scheduling System for Flexible Manufacturing. In Kusiak, A., editor, *Flexible Manufacturing Systems: Methods and Studies*, North-Holland, Amsterdam.

[11] Peterson, W. P. (1985). Diffusion Approximations for Networks of Queues with Multiple Customer Types. Unpublished Ph.D. Thesis, Dept. of Operations Research, Stanford University.

[12] Reiman, M. I. (1983). Some Diffusion Approximations with State Space Collapse. *Proc. Intl. Seminar on Modeling and Performance Evaluation Methodology*, Springer-Verlag, Berlin.

[13] Schonberger, R. J. (1982). *Japanese Manufacturing Techniques.* Free Press.

[14] Solberg, J. J. (1977). A Mathematical Model of Computerized Manufacturing Systems. Paper presented at the 4th International Conference on Production Research, Tokyo.

[15] Stidham, Jr., S. (1985). Optimal Control of Admission to a Queueing System. *IEEE Trans. Aut. Cont.* AC-30, 705-713.

[16] Wein, L. M. (1986). Scheduling Semiconductor Wafer Fabrication. *Tech. Report for Semiconductor Research Corporation.*

[17] Wein, L. M. (1987). Asymptotically Optimal Scheduling of a Two-Station Multiclass Queueing Network. Ph. D. Thesis, Department of Operations Research, Stanford University.

[18] Whitt, W. (1971). Weak Convergence Theorems for Priority Queues: Preemptive-Resume Discipline. *J. Appl. Prob.* 8, 74-94.

10/14/88

Date Due