STABILIZING BATCH-PROCESSING NETWORKS

J. G. DAI

School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0205, dai@isye.gatech.edu

CAIWEI LI

Oracle, 300 Oracle Parkway, Room 1084, Redwood City, California 94065, caiwei.li@oracle.com

In a batch-processing network, multiple jobs can be formed into a batch to be processed in a single service operation. The network is multiclass in that several job classes may be processed at a server. Jobs in different classes cannot be mixed into a single batch. A batch policy specifies which class of jobs is to be served next. Throughput of a batch-processing network depends on the batch policy used. When the maximum batch sizes are equal to one, the corresponding network is called a standard-processing network, and the corresponding service policy is called a dispatch policy. There are many dispatch policies that have been proven to maximize the throughput in standard networks. This paper shows that any *normal* dispatch policy can be converted into a batch policy that preserves key stability properties. Examples of normal policies are given. These include static buffer priority (SBP), first-in–first-out (FIFO), and generalized round robin (GRR) policies.

Received February 2001; revision received August 2001; accepted December 2001.
Subject classifications: Production/scheduling: sequencing, stochastic. Queues: networks, batch/bulk.
Area of review: Stochastic Models.

1. INTRODUCTION

Our study involves batch-processing networks in which multiple jobs can be processed as a batch in a single service operation. The size of a batch is limited by the physical capacity of the server or by the number of jobs available. The processing time of a batch is independent of the size of the batch. A semiconductor wafer fabrication facility, known as a wafer fab, is an example of a batch-processing network. In a wafer fab, diffusion furnaces can often process up to a dozen jobs at a time. However, the processing time of a batch may be as long as eight hours, as much as 100 times longer than a typical processing step in other areas.

In a batch-processing network such as a wafer fab, product flows are reentrant. Multiple processing steps, called job classes in this paper, compete for service at a single service station. When a server is ready to load the next batch, the class of jobs to be loaded next must be determined. A policy specifying such decisions is called a batch policy. A common issue is whether a server should wait for a full batch in order to fully utilize the server's capacity.

This paper is concerned with the throughput or production rate in a batch-processing network. As discussed further at the end of this introduction, the throughput in such a network depends not only on the processing speeds of the servers, but also on the batch policy employed. We contend that throughput is a more important performance measure than utilization of each individual server. When a good throughput is achieved, the servers are automatically utilized at proper levels. Our research shows that to have a good throughput: (1) full batch classes should have high priority; (2) when there are no full batch classes at a station, it does not matter whether the server waits for a full batch; and (3) which full batch class is loaded next is important.

When there is no batch operation in a batch-processing network, we call the network a *standard-processing network*. Although a standard-processing network is in a special class of batch-processing networks, with maximum batch sizes being one, we call the corresponding service policy in the standard network a *dispatch policy*. There have been many dispatch policies that have been proven to maximize the throughput; see, for example, Kumar and Seidman (1990), Bramson (1996a, b), Kumar and Kumar (1996), Dai and Weiss (1996), and Chen and Zhang (1997a, b). In this paper, we present a general scheme for converting a dispatch policy into a batch policy. We prove that the corresponding batch policy preserves certain stability properties of the dispatch policy. In particular, a dispatch policy that maximizes the throughput in a standard network can be turned into a batch policy that maximizes the throughput in the corresponding batch-processing network.

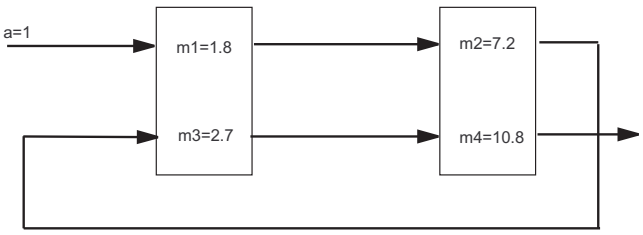
Most of the stability analyses in literature have been limited to standard-processing networks, also called *multiclass-queueing networks*, as advanced by Harrison (1988). Two exceptions are Maglaras and Kumar (1999) and Kumar and Zhang (2000), in which batch-processing networks were studied. In Maglaras and Kumar, a family of discrete review batch policies was shown to maximize the throughput. In Kumar and Zhang, a family of fluctuation-smoothing batch policies was shown to maximize the throughput in special networks called reentrant lines by Kumar (1993).

In the stability analysis for a standard-processing network, the standard tool is to use fluid models; see, Rybko and Stolyar (1992), Dai (1995), Stolyar (1995), Dai and Meyn (1995), Chene (1995), and Bramson (1998). Jennings (2000b) extended the fluid model tool for processing networks with setups. In this paper, as in Kumar and Zhang (2000), we also extend the fluid model tool to batch-processing networks.

0030-364X/03/5101-0123 $05.00
1526-5463 electronic ISSN

Operations Research © 2003 INFORMS
Vol. 51, No. 1, January–February 2003, pp. 123–136

Figure 1. A two-station, four-class batch-processing network.



The following is an example of a batch-processing network, illustrating that throughput depends on the batch policy employed. The network has two single-server service stations serving four job classes, as illustrated in Figure 1. Each job follows four processing steps, alternating between stations 1 and 2. Jobs being processed or waiting to be processed in step k are called class k jobs and reside in buffer k. The maximum batch sizes for servers 1 and 2 are 5 and 20, respectively. Jobs are assumed to arrive from the outside following a Poisson process with rate $\alpha = 1$ job per minute. The processing times for class k batches are independent, exponentially distributed with mean m_k, $k = 1, 2, 3, 4$. The mean service times are set to be $m_1 = 1.8$, $m_2 = 7.2$, $m_3 = 2.7$, and $m_4 = 10.8$ minutes, as shown in the figure. The traffic intensities for stations 1 and 2, to be defined in (2.5) in §2, are given by

$$\rho_1 = \alpha(m_1 + m_3)/5 = 0.9 \quad \text{and} \quad \rho_2 = \alpha(m_2 + m_4)/20 = 0.9.$$

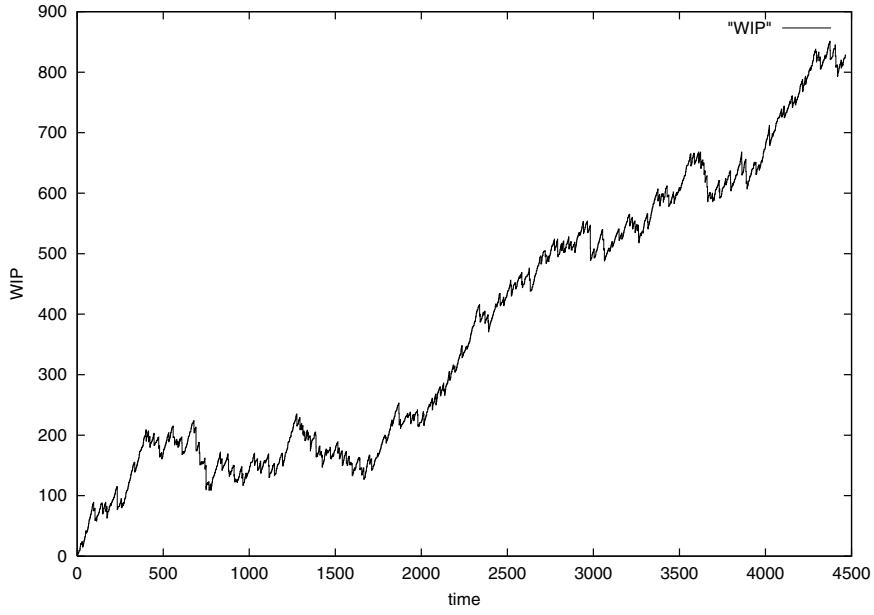
Therefore, the usual traffic condition (2.6) is satisfied for the parameter set. Intuitively, the batch-processing network should have enough capacity to handle all incoming jobs, achieving a throughput of 1 job per minute.

Since last-buffer-first-serve (LBFS) dispatch policy maximizes the throughput in a standard reentrant line (Dai and Weiss (1996), Kumar and Kumar (1996)), we employ the LBFS batch-policy in the batch-processing network. Under the LBFS policy, each server always loads the highest nonempty class to form a batch, even though the selected class may have only 1 job in it. We simulate this processing network by using the ASAP software package produced by AutoSimulations Inc. The following shows the average times in system:

Number of jobs leaving the system	50	500	5,000	50,000
Average time in system	54.2	208.4	1,057.3	6,831.6

Figure 2 plots the total number of jobs in the system as time increases. Clearly, the system is unstable, thus it cannot handle the offered load in the long-run. On the other hand, the same simulation shows that, after completing 50,000 jobs, server 1 is busy 96% of the time with average batch size 4.19 jobs and server 2 is busy 99.97% of the time with average batch size 16.41 jobs. The servers are apparently heavily utilized, yet the system is unstable. Under the LBFS batch policy, server 2 keeps serving class 4 batches that may have only 5 jobs, sent recently from class 3 by server 1, although class 2 has a large number of jobs waiting. This example shows that a naive implementation of a service policy may lead to an extremely inefficient system, although the policy performs well in a standard network. This source of inefficiency can be eliminated by employing the *full batch* policies to be described in §2. Under the full LBFS batch policy, server 2 gives high priority to class 2 when class 2 has a large number of jobs and class 4 has fewer than 20 jobs. Under this modified LBFS policy, the

Figure 2. The total number of jobs in system.



system can handle the offered load, achieving throughput of 1 job per minute.

The preceding source of inefficiency seems easy to identify and to correct. There is another source of inefficiency that is subtle and difficult to identify. This inefficiency occurs in processing networks having reentrant flows even when there are no batch operations. The challenge here is to decide which full batch class to load next when there are multiple full batch classes. Poor decisions lead to low utilization of servers, and at the same time the number of jobs in the system building up to infinity. Because this inefficiency phenomenon has been well studied in the literature (Kumar-Seidman 1990, Lu and Kumar 1991, Rybko and Stolyar 1992, Bramson 1994, and Seidman 1994), we refer readers to these papers for further discussion. (A more recent explanation can be found in Dai and Vande Vate 2000 and Hasenbein 1997 through virtual and pseudo stations.)

The rest of this paper is organized as follows. In §2, we introduce batch-processing networks and their corresponding standard-processing networks. We then describe a general scheme for converting a dispatch policy into a batch policy. We also define the notion of rate stability and present the main theorem of the paper. In §3, we introduce the fluid models of batch- and standard-processing networks. We establish that the stability of a fluid model implies the stability of the corresponding processing network, and introduce fluid limits that are used to justify fluid equations defining a fluid model. In §4, we study the relationship between batch- and standard-fluid models; through the relationship, we then define *normal dispatch policies* in a standard network, a key notion used in the statement of our main theorem. Finally, we present examples of normal dispatch policies in §5. These include static buffer priority, first-in–first-out, and generalized round robin policies. The paper concludes with a discussion of possible extensions in §6.

2. OPEN MULTICLASS BATCH-PROCESSING NETWORKS

In this section, we first introduce the open multiclass batch-processing networks, called batch-processing networks, that are the focus of this study. In a batch-processing network, multiple jobs can form a batch to be served in a single service operation. We then introduce their corresponding *standard-processing networks* that are identical to batch-processing networks except that jobs are processed one at a time. Finally, we describe a general mechanism of constructing an (induced) batch policy for the batch network from a dispatch policy for the standard network.

2.1. The Batch-Processing Network

The network under study has J single-server stations and K job classes. Stations are labeled by $j = 1, \dots, J$ and classes by $k, \ell = 1, \dots, K$. Class k jobs are served at a unique station $\sigma(k)$. For each station, more than one class might be served. Each station has an unlimited waiting space for each job class. Multiple jobs can form a *batch* that is to be processed in a single service operation. Each server always forms a batch as large as possible and the largest batch size for class k is B_k. We assume that jobs in different classes cannot be merged into a batch. The processing time for a batch is independent of the batch size.

Jobs arrive at the network from outside, and change classes as they move through the network. When a batch finishes its processing, it is split into individual jobs again, and these jobs are individually routed to the next class or outside. Each job eventually will leave the network. The ordered sequence of classes that a job visits in the network is called a route.

We use $\mathcal{C}(j)$ to denote the set of classes that belong to station j. When j and k appear together, we implicitly set $j = \sigma(k)$. For each class k, there are three groups of cumulative processes: $E_k = \{E_k(t), t \geqslant 0\}$, $V_k = \{V_k(n)\colon n = 1, 2, \dots\}$, and $\Phi^k = \{\Phi^k(n)\colon n = 1, 2, \dots\}$. For each time $t \geqslant 0$, $E_k(t)$ counts the number of external arrivals to class k in $[0, t]$. For each positive integer n, $V_k(n)$ is the total service time requirement for the first n batches (regardless of batch size) in class k. For each positive integer n, $\Phi^k(n)$ is a K-dimensional vector taking values in $\mathbb{Z}_+^K$. For each class ℓ, $\Phi_\ell^k(n)$ is the total number of jobs going to class ℓ among the first n jobs finishing services at class k. By convention, we assume

$$E_k(0) = 0, \quad V_k(0) = 0, \quad \text{and} \quad \Phi^k(0) = 0.$$

For each time $t \geqslant 0$, we extend the definitions of $V_k(t)$ and $\Phi^k(t)$ as

$$V_k(t) = V_k(\lfloor t \rfloor) \quad \text{and} \quad \Phi^k(t) = \Phi^k(\lfloor t \rfloor),$$

where $\lfloor t \rfloor$ denotes the largest integer less than or equal to t. We call (E, V, Φ) the primitive processes, where $E = \{E(t), t \geqslant 0\}$, $V = \{V(t), t \geqslant 0\}$, and $\Phi = \{\Phi(t), t \geqslant 0\}$, with $E(t) = (E_1(t), E_2(t), \dots, E_K(t))'$, $V(t) = (V_1(t), V_2(t), \dots, V_K(t))'$, and $\Phi(t) = (\Phi^1(t), \Phi^2(t), \dots, \Phi^K(t))'$. We assume that the strong law of large numbers holds for the primitive processes, namely, with probability one,

$$\lim_{t\to\infty} \frac{E_k(t)}{t} = \alpha_k, \quad \lim_{t\to\infty} \frac{V_k(t)}{t} = m_k, \quad \text{and}$$
$$\lim_{t\to\infty} \frac{\Phi(t)}{t} = P. \tag{2.1}$$

The parameter (α, m, P) with $\alpha = (\alpha_1, \dots, \alpha_K)'$ and $m = (m_1, \dots, m_K)'$ has the following natural interpretations: For each class k, α_k is the external job arrival rate at class k and m_k is the mean service time for class k batches. (Recall that the processing time of a batch is independent of its batch size.) For classes k and ℓ, $P_{k\ell}$ is the long-run fraction of class k jobs that become class ℓ. It is also called the routing probability from class k to class ℓ. The $K \times K$

matrix $P = (P_{k\ell})$ is called the routing matrix. We assume that the network is open, i.e., the matrix

$$Q = I + P' + (P')^2 + \cdots$$

is finite, which is equivalent to the fact that $(I - P')$ is invertible and $Q = (I - P')^{-1}$. A reentrant line is a special type of processing network in which all jobs follow a deterministic route of K stages, and jobs may visit some stations multiple times.

For future purposes, we introduce the counting process $S = \{S(t): t \geqslant 0\}$ associated with the primitive service process V. For each time $t \geqslant 0$, $S(t) = (S_1(t), \ldots, S_K(t))'$, with

$$S_k(t) = \max\{n: V_k(n) \leqslant t\}, \quad k = 1, 2, \ldots, K.$$

It follows from the strong law of large numbers (2.1) that

$$\lim_{t\to\infty} \frac{S_k(t)}{t} = \mu_k, \quad k = 1, \ldots, K, \tag{2.2}$$

where $\mu_k = 1/m_k$.

Whenever a server is ready to load a batch, it needs a policy to decide which batch to serve next. Such a policy is called a *batch policy*. We assume that, within a class, first-in-first-serve (FIFO) policy is used to form a batch. Once class k is selected by a server, the server always attempts to form a batch of size B_k if possible. Once a service is started, the service cannot be preempted. A class k with at least B_k jobs is called a *full batch class*. In this paper, we restrict ourselves to *full batch policies*. Namely, at the end of a service, the server has to load a full batch class when one is available at the station. When there is no full batch class at a station, the server can choose to idle. Waiting for additional jobs to form full batches is a common practice in some industries including wafer fabs. The full batch policies can and should be relaxed in some cases; see §6 for possible extensions.

2.2. The Standard-Processing Network

We now define the *standard-processing network* that corresponds to a batch-processing network. The standard network is identical to the batch-processing network except that (a) the maximum batch size is one, and (b) the primitive service process is given by $\tilde{V}_k = \{\tilde{V}_k(n): n = 1, \ldots\}$ where $\tilde{V}_k(n) = V_k(n)/B_k$ and B_k is the batch size of class k in the original network. As a result, the counting process $\tilde{S}$ associated with the primitive service process $\tilde{V}$ is described by $\tilde{S}_k(t) = \max\{n : \tilde{V}_k(n) \leqslant t\} = S_k(B_k t)$, $k = 1, 2, \ldots, K$, and the strong law of large numbers becomes

$$\lim_{n\to\infty} \frac{\tilde{V}_k(n)}{n} = m_k/B_k \quad \text{and} \quad \lim_{t\to\infty} \frac{\tilde{S}_k(t)}{t} = B_k\mu_k, \quad k = 1, \ldots, K. \tag{2.3}$$

In short, the standard network processes one job at a time, and when class k jobs are in service, the server speeds up by a factor of B_k over the service in the batch network.

For a batch-processing network driven by the primitive processes (E, V, Φ) with maximum batch sizes $(B_1, \ldots, B_K)'$, the corresponding standard-processing network is driven by the primitive processes $(E, \tilde{V}, \Phi)$ and the maximum batch sizes is one.

Because a standard network is a special case of a batch-processing network, a service or batch policy is also needed to operate such a network. We call a service policy in such a network a *dispatch policy*. The alternative term is needed to distinguish the batch policy introduced in the previous section. A major result of this paper is to use a dispatch policy to construct a corresponding batch policy that preserves the stability property of the dispatch policy. The construction will be carried out in the next section.

2.3. The Induced Batch Policy

In this section, we describe a procedure to construct the corresponding batch policy for a batch network from a dispatch policy for a standard network. Let π be a dispatch policy for the standard network.

We now define an induced batch policy $\tilde{\pi}$ for the batch network. The policy π dictates which nonempty class should be served next based on the system state of the corresponding standard network. In the batch network, any class k with fewer than B_k jobs is considered to be "empty." In other words, the system state component corresponding to class k is set at 0. Based on that revised state, each server in the batch-processing network uses the dispatch policy π to select a "nonempty" class ℓ to work on. Once class ℓ is selected according to policy π, the server serves exactly B_ℓ jobs of class ℓ in a single batch. If all classes at a station are "empty," the server employs any batch policy to select a job to work on, including idling. To be concrete, when a station is "empty," we still use π to pick a nonempty class to work on according to the original system state.

The goal of this paper is to show that the batch-processing network operating under batch policy $\tilde{\pi}$ is stable if the standard network operating under π is stable. (The stability definition will be given in §2.4 below.) In formulating our main theorem, Theorem 2.1, we need to restrict ourselves to a family of *normal* dispatch policies, whose precise definition will be given in §4. Most practical dispatch policies are normal. As an illustration, we prove in §5 that three families of dispatch policies are normal. They are static buffer priority (SBP), first-in-first-out (FIFO), and generalized round robin (GRR) policies.

2.4. Rate Stability and the Main Result

For both the batch- and standard-processing networks, the nominal total arrival rates and traffic intensities are identical. Let $\lambda = (\lambda_1, \ldots, \lambda_K)'$ be the vector of nominal total arrival rates (for both the batch- and standard-processing networks). It satisfies the following system of equations:

$$\lambda_l = \alpha_l + \sum_{k=1}^{K} \lambda_k P_{kl} \quad \text{for} \quad \ell = 1, 2, \ldots, K. \tag{2.4}$$

In vector form, $\lambda = \alpha + P'\lambda$. Since P is transient, the unique solution to (2.4) of λ is given by $\lambda = Q\alpha$. We define the traffic intensity ρ_j for server j (in both networks) as

$$\rho_j = \sum_{k \in \mathscr{C}(j)} \lambda_k (m_k / B_k), \quad j = 1, \ldots, J, \tag{2.5}$$

with ρ being the corresponding vector. Note that in the batch network, ρ_j is the nominal utilization of server j if every batch is of the maximum size. Because class k batch sizes can be smaller than B_k, the fraction of time that server j is busy may be greater than ρ_j in the batch network. When

$$\rho_j \leqslant 1, \quad j = 1, \ldots, J, \tag{2.6}$$

we say that the usual traffic condition is satisfied.

We now define the rate stability for a batch-processing network. Let $D_k(t)$ denote the number of jobs that have departed from class k in $[0, t]$ in the batch-processing network. In the following definition, the term *state* is used. The precise definition of a state depends on the particular batch policy used. It typically includes, but is not limited to, the number of jobs in each class, the remaining processing times and batch sizes of the batches that are being processed, and the remaining interarrival times for jobs arriving from outside. We do not attempt a precise definition here. Roughly speaking, a state is a snapshot of the network at any given time. It should contain enough information that once the current state of the network is given, the future evolution of the network is completely determined in distribution. Readers are referred to Dai (1995) and Bramson (1998) for examples and additional discussions of states in standard networks under various policies.

DEFINITION 2.1. The batch-processing network is *rate stable* if, for each fixed initial state with probability one,

$$\lim_{t \to \infty} \frac{D_k(t)}{t} = \lambda_k \quad \text{for } k = 1, \ldots, K. \tag{2.7}$$

The batch network is rate stable if the throughput rate or departure rate from a class is equal to the nominal total arrival rate to that class. Rate stability has been advanced by Stidham and his co-authors (see El-Taha and Stidham (1999) and the references there). This notion of stability was first introduced for multiclass queueing network settings in Chen (1995). As in a standard network, the usual traffic condition is necessary for rate stability of a batch-processing network (Dai 1999). There are other definitions of stability, such as positive Harris recurrence (Dai 1995). The results in this paper can be extended to those settings as well.

As mentioned before, the main result of this paper is that a dispatch policy of a standard network can be turned into a batch policy that shares a similar stability property. The precise form of the result is stated in the following theorem. The definitions of "normal policy" and "fluid model" used in the following theorem are delayed to later sections. The fluid model and its stability will be introduced in the next section. The definition of a normal policy will be introduced in §4.

THEOREM 2.1. *For a given batch-processing network, assume that a dispatch policy π is normal for the corresponding standard network. The batch-processing network operating under the induced batch policy $\tilde{\pi}$ is rate stable if the standard fluid model operating under π is weakly stable.*

The proof of Theorem 2.1 will be presented in §4. Section 5 is devoted to the applications of the theorem.

3. PROCESSING NETWORK AND FLUID MODEL EQUATIONS

In this section, we define fluid models corresponding to the batch- and standard-processing networks. Fluid models are continuous, deterministic analogs of batch- and standard-processing networks, and are defined through a set of equations. To describe the fluid models, we start with the dynamic equations for batch- and standard-processing networks. Unless explicitly stated otherwise, we assume that the batch-processing network is operated under a full batch policy $\tilde{\pi}$ and the standard-processing network is operated under a nonidling dispatch policy π.

3.1. Dynamics of Batch and Standard Networks

The dynamics of the batch network can be described by process $\mathbb{X} = (A, D, T, U, Y, Z)$. The components $A = \{A(t), t \geqslant 0\}$, $D = \{D(t), t \geqslant 0\}$, $T = \{T(t), t \geqslant 0\}$, and $Z = \{Z(t), t \geqslant 0\}$ are K-dimensional. For each class k, $A_k(t)$ denotes the number of jobs that have arrived to class k (from external and internal sources) in $[0, t]$, $D_k(t)$ denotes the number of jobs that have departed from class k in $[0, t]$, $T_k(t)$ denotes the amount of time that server $j = \sigma(k)$ has spent in serving class k batches during interval $[0, t]$, and $Z_k(t)$ denotes the number of jobs in class k that are buffered or being served at station j at time t. The processes A, D, T, and Z are called the arrival, departure, server allocation, and jobcount processes, respectively. The components $U = \{U(t), t \geqslant 0\}$ and $Y = \{Y(t), t \geqslant 0\}$ are J-dimensional. For each station j, $U_j(t)$ denotes the total number of jobs at station j that are buffered or being served at time t, and $Y_j(t)$ denotes the total amount of time that server j has been idle in the time interval $[0, t]$. The process Y is called the cumulative idle time process. One can check that $\mathbb{X} = (A, D, T, U, Y, Z)$ satisfies the following set of equations:

$$A(t) = E(t) + \sum_k \Phi^k(D(t)), \quad t \geqslant 0, \tag{3.1}$$

$$Z(t) = Z(0) + A(t) - D(t), \quad t \geqslant 0, \tag{3.2}$$

$$Z(t) \geqslant 0, \quad t \geqslant 0, \tag{3.3}$$

$$U(t) = CZ(t), \quad t \geqslant 0, \tag{3.4}$$

$$CT(t) + Y(t) = et, \quad t \geqslant 0, \tag{3.5}$$

$$Y_j(t) \text{ increases only when } Z_k(t) < B_k \text{ for each } k \in \mathscr{C}(j), \quad j = 1, \ldots, J, \tag{3.6}$$

$$\text{additional equations associated with the particular batch policy } \tilde{\pi}. \tag{3.7}$$

Here C is the constituency matrix defined as

$$C_{jk} = \begin{cases} 1 & \text{if } k \in \mathscr{C}(j), \\ 0 & \text{otherwise,} \end{cases}$$

and e denotes the J vector of all 1's. Because we assume that, within a class, the FIFO policy is used to form batches, we have the following additional equations: for $0 \leqslant t_1 < t_2$ and $k = 1, \ldots, K$,

$$S_k(T_k(t_2)) - S_k(T_k(t_1)) \leqslant \frac{1}{B_k}(D_k(t_2) - D_k(t_1) + B_k - 1) \tag{3.8}$$

when $Z_k(s) \geqslant B_k$ for $s \in [t_1, t_2]$, and

$$S_k(T_k(t_2)) - S_k(T_k(t_1)) \geqslant \frac{1}{B_k}(D_k(t_2) - D_k(t_1)). \tag{3.9}$$

To check (3.8), we note that the left side is the number of class k batches completed in $[t_1, t_2]$. Because there are enough jobs in class k throughout the time interval $[t_1, t_2]$, any class k batch formed in (t_1, t_2) has batch size B_k. However, if there is a class k batch in service time at t_1, this batch was formed before t_1 and whose size may be smaller than B_k. In any case, the right side of (3.8) provides an upper bound on the number of class k batches completed in $[t_1, t_2]$. Thus, inequality (3.8) holds. Inequality (3.9) can be justified similarly. We call Equations (3.1)–(3.9) batch network equations. We note that T and Y are continuous, and that A, D, and Z are right continuous with left limits. All variables are nonnegative in each component, with A, D, T, and Y being nondecreasing. By assumption,

$$A(0) = D(0) = T(0) = Y(0) = 0.$$

For each batch network driven by (E, V, Φ), the corresponding standard network driven by $(E, \tilde{V}, \Phi)$ has similar processes. To contrast with batch-network processes, they are denoted by $(\tilde{A}, \tilde{D}, \tilde{T}, \tilde{U}, \tilde{Y}, \tilde{Z})$. The equations governing these processes are the same as the ones for batch networks, except that Equations (3.8)–(3.9) are reduced to

$$\tilde{S}(\tilde{T}(t)) = \tilde{D}(t) \quad \text{for all } t \geqslant 0, \tag{3.10}$$

which is well known for standard networks operating under a head-of-line dispatch policy, and Equation (3.7) is replaced by

additional equations associated with the particular dispatch policy π. (3.11)

3.2. Batch and Standard Fluid Models

Let $\bar{\mathbb{X}} = (\bar{A}, \bar{D}, \bar{T}, \bar{U}, \bar{Y}, \bar{Z})$ be the continuous, deterministic analog of the batch network process $\mathbb{X} = (A, D, T, U, Y, Z)$. Its components satisfy the following equations:

$$\bar{A}(t) = \alpha' t + P' \bar{D}(t), \quad t \geqslant 0, \tag{3.12}$$

$$\bar{Z}(t) = \bar{Z}(0) + \bar{A}(t) - \bar{D}(t), \quad t \geqslant 0, \tag{3.13}$$

$$\bar{Z}(t) \geqslant 0, \quad t \geqslant 0, \tag{3.14}$$

$$\bar{U}(t) = C\bar{Z}(t), \quad t \geqslant 0, \tag{3.15}$$

$$C\bar{T}(t) + \bar{Y}(t) = et, \quad t \geqslant 0, \tag{3.16}$$

$\bar{Y}_j(t)$ increases only when $\bar{U}_j(t) = 0$, $j = 1, \ldots, J$, (3.17)

$$\bar{D}_k(t_2) - \bar{D}_k(t_1) \leqslant B_k \mu_k (\bar{T}_k(t_2) - \bar{T}_k(t_1)) \quad \text{for } 0 \leqslant t_1 < t_2, \quad k = 1, \ldots, K, \tag{3.18}$$

$$\bar{D}_k(t_2) - \bar{D}_k(t_1) = B_k \mu_k (\bar{T}_k(t_2) - \bar{T}_k(t_1)) \quad \text{if } \bar{U}_j(s) > 0 \ \forall s \in [t_1, t_2], \quad 0 \leqslant t_1 < t_2, \tag{3.19}$$

additional equations associated with the particular batch policy $\tilde{\pi}$. (3.20)

In (3.19), as before, j is set to be $s(k)$. Equations (3.12)–(3.20) are called batch fluid model equations, and they define the batch fluid model. Any process $\bar{\mathbb{X}} = (\bar{A}(t), \bar{D}(t), \bar{T}(t), \bar{U}(t), \bar{Y}(t), \bar{Z}(t))$ satisfying (3.12)–(3.20) is called a batch fluid model solution. Similarly, we can define the standard fluid model, which consists of the same set of fluid model equations except that Equations (3.18) and (3.19) are replaced by

$$\hat{D}_k(t) = B_k \mu_k \hat{T}_k(t) \quad \text{for all } t \geqslant 0, \quad k = 1, \ldots, K, \tag{3.21}$$

and (3.20) is replaced by

additional equations associated with the particular dispatch policy π. (3.22)

Any process $\hat{\mathbb{X}} = (\hat{A}, \hat{D}, \hat{T}, \hat{U}, \hat{Y}, \hat{Z})$ satisfying Equations (3.12)–(3.17) and Equations (3.21)–(3.22) is called a standard fluid model solution.

Definition 3.1. A batch fluid model is said to be *weakly stable* if for each batch fluid model solution $\bar{\mathbb{X}}$ with $\bar{Z}(0) = 0$, $\bar{Z}(t) = 0$ for $t \geqslant 0$.

Weak stability of a standard fluid model can be defined similarly as in Chen (1995).

3.3. Connection Between Processing Networks and Fluid Models

The criterion for including an equation in the batch or standard fluid model is that the equation is satisfied by *fluid limits*. A fluid limit of a batch-processing network is obtained through a law-of-large-number procedure on the batch network process. Note that the batch network process $\mathbb{X}$ is random, depending on the sample ω in an underlying probability space. To denote such dependence explicitly, we sometimes use $\mathbb{X}(\omega)$ to denote the batch network process with sample ω. For an integer d, $\mathbb{D}^d[0, \infty)$ denotes the set of functions x: $[0, \infty) \to \mathbb{R}^d$ that are right continuous on $[0, \infty)$ and have left limits on $(0, \infty)$. An element x in $\mathbb{D}^d[0, \infty)$ is sometimes denoted by $x(\cdot)$ to emphasize that x is a function of time. For each ω, $\mathbb{X}(\omega)$ is an element in $\mathbb{D}^{4K+2J}[0, \infty)$.

For each $r > 0$, define

$$\bar{\mathbb{X}}^r(t,\omega) = r^{-1}\mathbb{X}(rt,\omega), \quad t \geqslant 0. \tag{3.23}$$

Note that again for each $r > 0$, $\bar{X}^r(\cdot,\omega)$ is an element in $\mathbb{D}^{4K+2J}[0,\infty)$. The scaling in (3.23) is called the fluid or law-of-large-number scaling.

DEFINITION 3.2. A function $\bar{\mathbb{X}} \in \mathbb{D}^{4K+2J}[0,\infty)$ is said to be a *fluid limit* of the batch-processing network if there exists a sequence $r_n \to \infty$ and a sample ω satisfying (2.1) such that

$$\lim_{n\to\infty} \bar{\mathbb{X}}^{r_n}(\cdot,\omega) \to \bar{\mathbb{X}}(\cdot),$$

where, here and later, the convergence is interpreted as the uniform convergence on compact sets (u.o.c.).

The existence of fluid limits is well known. A standard argument like the one in Dai (1995) shows that for any $r \to \infty$ and any sample ω, there is a sequence r_n such that $\bar{T}^{r_n}(\cdot,\omega)$ converges as $n \to \infty$. Fix an ω that satisfies (2.1). The convergence of $\bar{T}^{r_n}$, together with Equation (3.9) and condition (2.1), implies that $\bar{D}^{r_n}$ converges. This latter convergence, together with Equation (3.1) and condition (2.1), implies that $\bar{A}^{r_n}$ converges. The convergence of other components of $\bar{\mathbb{X}}^{r_n}$ then readily follows. Thus, $\bar{\mathbb{X}}^{r_n}$ converges to a fluid limit as $n \to \infty$.

PROPOSITION 3.1. *Each fluid limit of the batch-processing network operating under a full batch policy $\tilde{\pi}$ is a fluid model solution to the batch fluid model.*

PROOF. Let $\bar{\mathbb{X}}$ be a fluid limit. Equation (3.18) follows from (3.9). To prove (3.19), it is enough to show that for each s such that $\bar{U}_j(s) > 0$ and $\bar{\mathbb{X}}$ is differential at s,

$$\dot{\bar{D}}_k(s) = B_k \mu_k \dot{\bar{T}}_k(s), \tag{3.24}$$

where, for a function f, $\dot{f}(s)$ denotes the derivative of f at s. To prove (3.24), let $r_n \to \infty$ be a sequence such that $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ as $n \to \infty$. Since $\bar{U}_j(s) > 0$, there exists a class ℓ at station j such that $\bar{Z}_\ell(s) > 0$. By the continuity of $\bar{Z}$, there exists a $\delta > 0$ such that $\min_{t\in[s-\delta,s+\delta]} \bar{Z}_\ell(t) > 0$. Because $\bar{Z}^{r_n}(\cdot) \to \bar{Z}(\cdot)$ u.o.c., similar to the derivation of (5.3) in the proof of Proposition 5.1, one has that for large enough n,

$$Z_\ell(u) \geqslant B_\ell \quad \text{for } u \in [r_n(s-\delta), r_n(s+\delta)].$$

Now, fix a class k at station j. Because class ℓ can always form full batches in $[r_n(s-\delta), r_n(s+\delta)]$, any class k batch formed during $[r_n(s-\delta), r_n(s+\delta)]$ has to be a full batch as well. Thus, (3.8) holds for $t_1 = r_n(s-\delta)$ and $t_2 = r_n(s+\delta)$. Namely,

$$\begin{aligned} &S_k(T_k(r_n(s+\delta))) - S_k(T_k(r_n(s-\delta))) \\ &\quad \leqslant \frac{1}{B_k}(D_k(r_n(s+\delta)) - D_k(r_n(s-\delta)) + B_k - 1). \end{aligned}$$

Dividing both sides of the preceding inequality by r_n and letting $n \to \infty$, one has

$$\mu_k(\bar{T}_k(s+\delta) - \bar{T}_k(s-\delta)) \leqslant \frac{1}{B_k}(\bar{D}_k(s+\delta) - \bar{D}_k(s-\delta)).$$

Dividing both sides of the preceding inequality by δ and letting $\delta \downarrow 0$, one has

$$\mu_k \dot{\bar{T}}_k(s) \leqslant \frac{1}{B_k}\dot{\bar{D}}_k(s). \tag{3.25}$$

Equation (3.24) now follows from (3.25) and (3.18). Other fluid model equations can be verified as in Dai (1995). □

THEOREM 3.2. *Let a batch policy $\tilde{\pi}$ be fixed. If the batch fluid model is weakly stable, then the corresponding batch-processing network is rate stable.*

PROOF. The theorem was first explicitly stated in Chen (1995) for the standard-processing networks. The proof of our theorem is identical to one for the standard network. See, for example, Dai (1999). □

4. CONNECTION BETWEEN STANDARD AND BATCH FLUID MODELS

Let $\bar{\mathbb{X}} = (\bar{A}, \bar{D}, \bar{T}, \bar{U}, \bar{Y}, \bar{Z})$ be a batch fluid model solution. We would like to convert it into a standard fluid model solution, $\hat{\mathbb{X}} = (\hat{A}, \hat{D}, \hat{T}, \hat{U}, \hat{Y}, \hat{Z})$. We define $\hat{\mathbb{X}}$ as follows: for each $t \geqslant 0$,

$$\hat{A}(t) = \bar{A}(t), \tag{4.1}$$

$$\hat{D}(t) = \bar{D}(t), \tag{4.2}$$

$$\hat{T}_k(t) = \frac{m_k}{B_k}\hat{D}_k(t), \quad k = 1, 2, \ldots, K, \tag{4.3}$$

$$\hat{Y}_j(t) = t - \sum_{k\in\mathscr{C}(j)} \hat{T}_k(t), \quad j = 1, \ldots, J, \tag{4.4}$$

$$\hat{U}(t) = \bar{U}(t), \tag{4.5}$$

$$\hat{Z}(t) = \bar{Z}(t). \tag{4.6}$$

PROPOSITION 4.1. *The $\hat{\mathbb{X}}$ constructed from $\bar{\mathbb{X}}$ by* (4.1)–(4.6) *satisfies standard fluid model Equations* (3.12)–(3.17) *and* (3.21).

PROOF. Because $\hat{Z}(t) = \bar{Z}(t)$, $\hat{A}(t) = \bar{A}(t)$, $\hat{U}(t) = \bar{U}(t)$, and $\hat{D}(t) = \bar{D}(t)$, and because $\bar{Z}(t)$, $\bar{A}(t)$, $\bar{U}(t)$, and $\bar{D}(t)$ satisfy Equations (3.12)–(3.15), $\hat{Z}(t)$, $\hat{A}(t)$, $\hat{U}(t)$, and $\hat{D}(t)$ also satisfy (3.12)–(3.15). By (4.4), Equation (3.16) is automatically satisfied. Since $\hat{D}(t)$ is nondecreasing, $\hat{T}(t)$ is also nondecreasing. To show that $\hat{Y}_j$ is nondecreasing, we note that for any $0 \leqslant t_1 < t_2$,

$$\hat{Y}_j(t_2) - \hat{Y}_j(t_1) = t_2 - t_1 - \left(\sum_{k\in\mathscr{C}(j)} \hat{T}_k(t_2) - \sum_{k\in\mathscr{C}(j)} \hat{T}_k(t_1)\right).$$

By definitions (4.2)–(4.3), we have

$$\begin{aligned} \hat{T}_k(t_2) - \hat{T}_k(t_1) &= \frac{m_k}{B_k}(\hat{D}_k(t_2) - \hat{D}_k(t_1)) \\ &= \frac{m_k}{B_k}(\bar{D}_k(t_2) - \bar{D}_k(t_1)) \leqslant \bar{T}_k(t_2) - \bar{T}_k(t_1), \end{aligned}$$

where the inequality follows from (3.18). Thus, we have

$$\hat{Y}_j(t_2) - \hat{Y}_j(t_1) \geqslant t_2 - t_1 - \left(\sum_{k \in \mathscr{C}(j)} \bar{T}_k(t_2) - \sum_{k \in \mathscr{C}(j)} \bar{T}_k(t_1) \right) \geqslant 0.$$

The last inequality follows from (3.16) and the fact that $\bar{Y}_j(\cdot)$ is nondecreasing. Thus, $\hat{Y}_j(t)$ is nondecreasing. Moreover, $\hat{Y}_j(t_2) - \hat{Y}_j(t_1) = 0$ when $\hat{U}_j(t) > 0$ for $t \in [t_1, t_2]$ because (3.19), (3.16), and (3.17) are satisfied for $\bar{\mathbb{X}}$. Thus, Equation (3.17) is also satisfied for $\hat{\mathbb{X}}$. By (4.3), Equation (3.21) is also true. □

We hope that $\hat{\mathbb{X}}$ also satisfies standard fluid model Equation (3.22). This, of course, depends on the particular dispatch policy used. As will be shown in the next section, $\hat{\mathbb{X}}$ satisfies (3.22) for many policies including static buffer priority, first-in–first-out, and generalized round robin policies. Anticipating the future growth of the list of dispatch policies, we define the notion of *normal* policy as follows.

Definition 4.1. A dispatch policy π is called *normal* if for any batch fluid model solution $\bar{\mathbb{X}}$ under batch policy $\tilde{\pi}$ induced from π, $\hat{\mathbb{X}}$ constructed by (4.1)–(4.6) also satisfies (3.22).

Proposition 4.2. *If a dispatch policy π operating in a standard-processing network is normal, then the batch fluid model under the induced batch policy $\tilde{\pi}$ is weakly stable if the standard fluid model under policy π is weakly stable.*

Proof. Let $\bar{\mathbb{X}}$ be any batch fluid model solution with $\bar{Z}(0) = 0$ under batch policy $\tilde{\pi}$. One can construct $\hat{\mathbb{X}}$ by (4.1)–(4.6). By Proposition 4.1 and the definition of a normal policy, $\hat{\mathbb{X}}$ is a standard fluid model solution under dispatch policy π. Because the standard fluid model is weakly stable and $\hat{Z}(0) = 0$, $\hat{Z}(t) = 0$ for $t \geqslant 0$. But $\bar{Z}(t) = \hat{Z}(t)$ for $t \geqslant 0$. Hence, we have $\bar{Z}(t) = 0$ for all $t \geqslant 0$. Thus, the batch fluid model under policy $\tilde{\pi}$ is weakly stable. □

With this preparation, the proof of Theorem 2.1 follows trivially.

Proof of Theorem 2.1. Assume that π is a normal dispatch policy in the standard network. Assume further that the corresponding standard fluid model is weakly stable. By Proposition 4.2, the batch fluid model operating under the induced batch policy $\tilde{\pi}$ is weakly stable. Theorem 2.1 then follows from Theorem 3.2. □

In the batch and standard fluid models, Equations (3.20) and (3.22) are determined by the batch policy and dispatch policy employed. Examples of these equations will be studied in the next section for SBP, FIFO, and GRR policies. Recall that $\bar{T}$ and $\hat{T}$ are server allocation processes for the batch and standard fluid models. Their derivatives $\dot{\bar{T}}(t)$ and $\dot{\hat{T}}(t)$ at time t indicate the instantaneous server allocation efforts among various classes. Thus, Equation (3.22) often involves $\dot{\bar{T}}(t)$ and (3.20) often involves $\dot{\hat{T}}(t)$. The following proposition is often useful to check that a dispatch policy is normal.

Proposition 4.3. *Let $\bar{\mathbb{X}}$ be a standard fluid model solution and $\hat{\mathbb{X}}$ be constructed from $\bar{\mathbb{X}}$ by* (4.1)–(4.6). *Then, for* $k = 1, \ldots, K$,

$$\dot{\hat{T}}_k(t) = \dot{\bar{T}}_k(t)$$

at each time t such that $\bar{\mathbb{X}}$ is differentiable at t and $\bar{U}_j(t) > 0$, where, as always, $j = \sigma(k)$.

Proof. Assume that $\bar{U}_j(t) > 0$. Since $\bar{\mathbb{X}}(t)$ is a continuous function of time t, there exists a $\delta > 0$ such that $\bar{U}_j(s) > 0$ for $s \in [t - \delta, t + \delta]$. It follows from (3.19) and (4.2)–(4.3) that we have

$$\hat{T}_k(t_2) - \hat{T}_k(t_1) = \bar{T}_k(t_2) - \bar{T}_k(t_1)$$

for any t_1 and t_2 with $t - \delta < t_1 < t_2 < t + \delta$, thus proving the proposition. □

5. EXAMPLES OF NORMAL POLICIES

In this section, we prove several dispatch policies that are normal. The policies, including static buffer priority (SBP), first-in-first-out (FIFO), and generalized round robin (GRR), have been extensively studied in the literature; see, for example, Bramson (1996a), Chen and Zhang (1997a, 2000) and Dai (1999). Our Theorem 2.1 shows that their corresponding induced batch policies preserve the stability property in batch-processing networks. Recall that all batch policies are assumed to be nonpreemptive, i.e., once a service is started, the server has to finish the service.

5.1. Static Buffer Priority Policies

Under a static buffer priority (SBP) dispatch policy, classes within a standard network are ranked. Higher ranked classes have higher priorities. Such an SBP policy can be denoted by a permutation π among classes. Let $\pi(k)$ indicate the priority of class k. If $\pi(k) > \pi(\ell)$, class k has higher priority than class ℓ.

For the SBP policy π, its induced batch policy is operated in the batch-processing network as follows: If $\pi(k) > \pi(\ell)$ and class k has at least B_k jobs, then class k has higher priority than class ℓ. If $\pi(k) > \pi(\ell)$, but class k has fewer than B_k jobs and class ℓ has at least B_ℓ jobs, then class ℓ has higher priority than class k because class k is treated as "empty."

Let $H_k = \{\ell\colon \ell \in \mathscr{C}(j), \pi(\ell) \geqslant \pi(k)\}$ denote the set of classes whose priorities are at least as high as class k. Let $\hat{\mathbb{X}}$ be a standard fluid model solution. Define

$$\hat{T}_k^+(t) = \sum_{\ell \in H_k} \hat{T}_\ell(t)$$

as the cumulative time that server $j = \sigma(k)$ has spent on all classes whose priorities are at least as high as class k. Define $\hat{Z}_k^+(t)$ similarly. It follows from Dai and Weiss (1996) that the standard fluid model Equation (3.22) takes the form

$$\dot{\hat{T}}_k^+(t) = 1 \quad \text{for each time } t \text{ such that } \hat{Z}_k^+(t) > 0 \text{ and } \hat{\mathbb{X}} \text{ is differential at time } t. \tag{5.1}$$

The batch fluid model Equation (3.20) under the induced batch policy $\tilde{\pi}$ is identical to that of a standard fluid model. The justification through fluid limits is the same as the

one in a standard network. For completeness, we provide a proof of Proposition 5.1 at the end of this section.

PROPOSITION 5.1. *Each fluid limit $\bar{\mathbb{X}}$ of the batch-processing network operating under an induced SBP policy satisfies the following equations: for $k = 1, \ldots, K$,*

$$\dot{\bar{T}}_k^+(t) = 1 \quad \text{for each time } t \text{ such that } \bar{Z}_k^+(t) > 0 \text{ and } \bar{\mathbb{X}} \text{ is differential at time } t. \tag{5.2}$$

The main result of this section is the following proposition.

PROPOSITION 5.2. *Any SBP dispatch policy is normal.*

PROOF. Let $\bar{\mathbb{X}}$ be a solution to the batch fluid model operating under the induced batch SBP policy. Let $\hat{\mathbb{X}}$ be a fluid solution constructed from $\bar{\mathbb{X}}$ by (4.1)–(4.6). Assume that $\hat{Z}_k^+(t) > 0$ and $\hat{\mathbb{X}}$ is differential at time t. From Proposition 4.3 and (5.2), we have

$$\dot{\hat{T}}_k^+(t) = \dot{\bar{T}}_k^+(t) = 1.$$

Thus, $\hat{\mathbb{X}}$ satisfies (5.1) and, therefore, is a solution to the standard fluid model. It follows from Definition 4.1 that the SBP dispatch policy is normal. □

A batch-processing reentrant line is a special batch-processing network. Classes can be arranged so that $\alpha_k = 0$ for $k = 2, \ldots, K$, and $P_{k,k+1} = 1$ for $k = 1, \ldots, K-1$. Two SPB dispatch policies: last-buffer-first-serve (LBFS) and first-buffer-first-serve (FBFS), have been studied in the literature. Under the LBFS policy, classes in later stages have higher priorities. Under the FBFS policy, classs in earlier stages have higher priorities.

COROLLARY 5.3. *A batch-processing reentrant line operating under either the induced LBFS or induced FBFS batch policy is rate stable whenever the usual traffic condition is satisfied.*

PROOF. Assume the usual traffic condition. It follows from Dai and Weiss (1996) and Kumar and Kumar (1996) that the standard fluid model is weakly stable under FBFS and LBFS dispatch policies. Because these policies are normal, the corollary follows from Theorem 2.1. □

PROOF OF PROPOSITION 5.1. Let $\bar{\mathbb{X}}$ be a fluid limit of the batch-processing network operating under the induced SBP policy. Let $r_n \to \infty$ be a corresponding sequence such that $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ as $n \to \infty$. Let $t > 0$ be fixed. Assume that $\bar{\mathbb{X}}$ is differential at time t and $\bar{Z}_k^+(t) > 0$. To prove (5.2), it is enough to show that $\dot{\bar{Y}}_k^+(t) = 0$, where $\bar{Y}_k^+(t) = t - \bar{T}_k^+(t)$ is the cumulative time that server $j = \sigma(k)$ can spend on classes outside H_k in $[0, t]$. Because $\bar{Z}_k^+(t) > 0$, by the continuity of $\bar{\mathbb{X}}$, there exists a $\delta > 0$ such that $\epsilon = \min_{t-\delta<s<t+\delta} \bar{Z}_k^+(s) > 0$. We would like to show that $\bar{Y}_k^+(t+\delta) - \bar{Y}_k^+(t-\delta) = 0$, from which the proposition follows.

Because $\bar{Z}^{r_n}(\cdot) \to \bar{Z}(\cdot)$ uniformly on compact sets (u.o.c.), and $r_n \to \infty$ as $n \to \infty$, there exists an N such that for all $n > N$,

$$\sup_{0 \leqslant s \leqslant t+\delta} |Z_k^+(r_n s)/r_n - \bar{Z}_k^+(s)| < \epsilon/2 \quad \text{and}$$

$$r_n \epsilon/2 \geqslant |\mathscr{C}(j)| \max_{\ell \in \mathscr{C}(j)} B_\ell,$$

where $Z_k^+(t)$ is the total number of jobs in H_k at time t and $|\mathscr{C}(j)|$ is the number of classes at station j. Hence $Z_k^+(r_n s) > r_n \epsilon/2$ for $n > N$ and $s \in (t-\delta, t+\delta)$, or equivalently, $Z_k^+(s) \geqslant r_n \epsilon/2$ for $s \in (r_n t - r_n \delta, r_n t + r_n \delta)$. Because $r_n \epsilon/2 \geqslant |\mathscr{C}(j)| \max_{\ell \in \mathscr{C}(j)} B_\ell$, for each $s \in (r_n t - r_n \delta, r_n t + r_n \delta)$, there exists an $\ell \in H_k$ such that

$$Z_\ell(s) \geqslant B_\ell. \tag{5.3}$$

Because the induced SBP batch policy is employed, in time interval $(r_n t - r_n \delta, r_n t + r_n \delta)$, it follows from (5.3) that server $j = \sigma(k)$ will not work on any classes that are not in H_k, except during the initial service period covering time instant $r_n t - r_n \delta$. It is possible for the server to continue working on a low priority class that is not in H_k because preemption is not allowed. (The server must be busy at time $r_n t - r_n \delta$ because there are enough jobs at the station at that time.) Let R_n be the remaining service time for the batch that is currently in service at time $r_n t - r_n \delta$. We have $Y_k^+(r_n t + r_n \delta) - Y_k^+(r_n t - r_n \delta) \leqslant R^n$ for $n > N$, where $Y_k^+(s) = \sum_{\ell \in H_k} Y_\ell(s)$ is the cumulative time that server $j = \sigma(k)$ can spend on classes that are not in H_k in $[0, s]$ in the batch-processing network.

Recall that $S_\ell(T_\ell(r_n t - r_n \delta))$ is the number of class ℓ batches completed by time $r_n t - r_n \delta$. If class ℓ is currently in service at time $r_n t - r_n \delta$, the server is working on the $(S_\ell(T_\ell(r_n t - r_n \delta)) + 1)$th batch. The total time for server j to finish all $S_\ell(T_\ell(r_n t - r_n \delta)) + 1$ batches is $V_\ell(S_\ell(T_\ell(r_n t - r_n \delta)) + 1)$. But the server has already spent $T_\ell(r_n t - r_n \delta)$ amount of time on class ℓ. Thus, the remaining processing time is equal to

$$V_\ell(S_\ell(T_\ell(r_n t - r_n \delta)) + 1) - T_\ell(r_n t - r_n \delta),$$

provided that class ℓ is in service at time $r_n t - r_n \delta$. Thus, we have

$$\frac{1}{r_n}[Y_k^+(r_n t + r_n \delta) - Y_k^+(r_n t - r_n \delta)]$$

$$\leqslant \max_{\ell \in \mathscr{C}(j)} \frac{V_\ell(S_\ell(T_\ell(r_n(t-\delta))) + 1) - T_\ell(r_n(t-\delta))}{r_n}$$

for $n > N$. Because $\bar{T}^{r_n}(\cdot) \to \bar{T}(\cdot)$, and (2.1) and (2.2) hold, we have

$$\lim_{n \to \infty} \frac{V_\ell(S_\ell(T_\ell(r_n(t-\delta))) + 1)}{r_n} = \bar{T}_\ell(t-\delta)$$

and

$$\lim_{n \to \infty} \frac{T_\ell(r_n(t-\delta))}{r_n} = \bar{T}_\ell(t-\delta).$$

Taking $n \to \infty$, we have

$$\bar{Y}_k^+(t+\delta) - \bar{Y}_k^+(t-\delta) \leqslant 0.$$

Because $\bar{Y}_k^+(\cdot)$ is nondecreasing, we have $\bar{Y}_k^+(t+\delta) - \bar{Y}_k^+(t-\delta) \geqslant 0$. Hence $\bar{Y}_k^+(t+\delta) - \bar{Y}_k^+(t-\delta) = 0$, thus $\dot{\bar{Y}}_k^+(t) = 0$, proving $\dot{\bar{T}}_k^+(t) = 1$. □

5.2. First-In–First-Out Policy

In a standard network, under the first-in-first-out (FIFO) dispatch policy, a server always picks a class whose head-of-line job arrived at its station earliest. The induced batch policy, called FIFO, works as follows in a batch-processing network. Whenever a server looks for a new class to load, it chooses the class, among the full batch classes, whose head-of-line job reached the station earliest. If there is no full batch class at the station, the server picks a class whose head-of-line job reached the station earliest. Thus, in a batch network operating under the FIFO policy, a server does not serve jobs according to a strict FIFO policy. The oldest job at a station may have to wait for more jobs in its class to arrive to form a full batch.

For the standard FIFO fluid model, the additional Equation (3.22) takes the form

$$\hat{D}_k(t+\hat{W}_j(t)) = \hat{A}_k(t), \quad k = 1, \dots, K, \tag{5.4}$$

for all $t > 0$, where

$$\hat{W}_j(t) = \sum_{k\in\mathscr{C}(j)} \frac{m_k}{B_k}\hat{Z}_k(t), \quad j = 1, \dots, J. \tag{5.5}$$

See, for example, Bramson (1996a).

For the batch FIFO fluid model, the additional fluid model Equation (3.20) takes the same form as in (5.4) and (5.5). This is the content of our next proposition.

Proposition 5.4. *Let $\bar{\mathbb{X}}$ be a fluid limit of the batch-processing network operating under the FIFO batch policy. It satisfies the following equations*:

$$\bar{D}_k(t+\bar{W}_j(t)) = \bar{A}_k(t), \quad k = 1, \dots, K, \tag{5.6}$$

for all $t > 0$, where

$$\bar{W}_j(t) = \sum_{k\in\mathscr{C}(j)} \frac{m_k}{B_k}\bar{Z}_k(t), \quad j = 1, 2, \dots, J. \tag{5.7}$$

We delay the proof until the end of this section.

Proposition 5.5. *The FIFO dispatch policy is normal.*

Proof. Let $\bar{\mathbb{X}}$ be a batch fluid model solution under the FIFO batch policy. Namely, $\bar{\mathbb{X}}$ satisfies Equations (3.12)–(3.19) and (5.6)–(5.7). Let $\hat{\mathbb{X}}$ be a fluid solution constructed from $\bar{\mathbb{X}}$ by (4.1)–(4.6). One can check that $\hat{\mathbb{X}}$ satisfies Equations (5.4)–(5.5). Thus, by Proposition 4.1, $\hat{\mathbb{X}}$ is a standard fluid model solution under the FIFO dispatch policy. Therefore, the FIFO dispatch policy is normal. □

A standard network is of Kelly type if, for each station, the mean processing times for all classes at a station are the same. Here we extend this definition to batch-processing networks. A batch-processing network is said to be of Kelly type if $B_k\mu_k$ are the same for all classes k at each station.

Corollary 5.6. *Assume that the usual traffic condition is satisfied in a FIFO batch-processing network of Kelly type. The batch network is rate stable.*

Proof. It was proven in Bramson (1996a) that the standard FIFO fluid model of Kelly type is weakly stable under the usual traffic condition. Because the FIFO dispatch policy is normal, the corollary follows from Theorem 2.1. □

Proof of Proposition 5.4. For the standard FIFO processing network,

$$D_k(t+W_j(t)) = Z_k(0) + A_k(t), \quad k = 1, \dots, K, \tag{5.8}$$

where $W_j(t)$ is the (immediate) workload at station j at time t, from which standard fluid model Equations (5.4) and (5.5) are derived. (See Harrison and Nguyen 1993 and Bramson 1996a).

For a batch-processing network, the definition of immediate workload for a server needs to be properly defined. Similar to the definition in a standard network, we define $W_j(t)$ to be the amount of total processing time that server j needs to spend to finish all the jobs that are currently at the station, assuming no more arrivals are allowed to the station after t. We now would like to establish a relationship that is analogous to (5.8). Two inequalities will be presented, one upper bound and the other lower bound. To explain these bounds, we take a closer look at time interval $[t, t+W_j(t)]$. Recall that there are $U_j(t)$ jobs at station j at time t. Some of these jobs (first type) are currently in service. Some (second type) will be served full batch with other jobs that are *currently* at the station. The remaining ones (third type) will be served either nonfull batch or together with jobs that arrive after time t. Note that it is possible for a job that arrives after time t to be processed before type 2 jobs. This job necessarily joins a batch with type 3 jobs, taking advantage of the early arrival of a type 3 job. The lower bound is given by

$$Z_k(0) + A_k(t) - B_k < D_k(t+W_j(t)) \quad \text{for } t \geqslant 0, \quad k = 1, \dots, K. \tag{5.9}$$

This bound follows from the fact that by time $t+W_j(t)$, all first and second types of jobs have left. To describe the upper bound, we let $\tau_j(t)$ be the total processing time of type 3 jobs. We claim that

$$D_k(t+W_j(t)-\tau_j(t)) < Z_k(0) + A_k(t) + B_k, \quad k = 1, \dots, K. \tag{5.10}$$

To check (5.10), in $[t, t+W_j(t)-\tau_j(t)]$, the server j cannot process more than $Z_k(t)+B_k$ class jobs. Thus, we have (5.10).

Assume that $\bar{\mathbb{X}}^{r_n}$ converges to a fluid limit $\bar{\mathbb{X}}$ as $n \to \infty$. To show that $\bar{\mathbb{X}}$ satisfies (5.6) and (5.7), because of (5.9) and (5.10) it suffices to show that

$$\bar{W}_j^{r_n}(\cdot) \to \sum_{k\in\mathscr{C}(j)} \frac{m_k}{B_k}\bar{Z}_k(\cdot) \tag{5.11}$$

and

$$\bar{\tau}_j^{r_n}(\cdot) \to 0, \tag{5.12}$$

where for $r > 0$, $\bar{W}^r(t) = W(rt)/r$ and $\bar{\tau}^r(t) = \tau(rt)/r$.

Let time $t \geqslant 0$ be fixed. Let $F_k(t)$ be the number of class k batches that can be formed from $Z_k(t)$ jobs that are at station j at time t. If class k is currently not in service, one can check that $F_k(t) = \lceil Z_k(t)/B_k \rceil$ in this case, where $\lceil x \rceil$ is the smallest integer that is as big as a nonnegative number x. If class k is currently in service, $F_k(t) - 1$ is the number of class k batches that can be formed from remaining jobs that are in class k at time t, excluding those currently in service. Thus, $F_k(t) = 1 + \lceil (Z_k(t) - \delta_k(t))/B_k \rceil$, where $\delta_k(t)$ is the size of the batch that is currently in service at time t. By our definition of immediate workload, we have

$$W_j(t) = \sum_{k \in \mathscr{C}(j)} V_k\left(S_k(T_k(t)) + F_k(t)\right) - t + Y_j(t). \tag{5.13}$$

So

$$\bar{W}_j^{r_n}(t) = \sum_{k \in \mathscr{C}(j)} \bar{V}_k^{r_n}\left(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}^{r_n}(t)\right) - t + \bar{Y}_j^{r_n}(t)$$

where, as usual, $\bar{V}^{r_n}(\cdot)$, $\bar{F}^{r_n}(\cdot)$, and $\bar{F}^{r_n}(\cdot)$ are fluid scalings of $V(\cdot)$, $S(\cdot)$, and $F(\cdot)$, respectively. Because $\bar{Z}_k^{r_n}(\cdot) \to \bar{Z}_k(\cdot)$ and $\delta_k(t) \leqslant B_k$ for all $t \geqslant 0$, we have $\bar{F}_k^{r_n}(\cdot) \to \bar{Z}_k(\cdot)/B_k$. As before, the uniform convergence on compact sets (u.o.c.) is used.

Because as $n \to \infty$, $\bar{S}_k^{r_n}(\cdot) \to \bar{S}_k(\cdot)$, $\bar{V}_k^{r_n}(\cdot) \to \bar{V}_k(\cdot)$, $\bar{T}_k^{r_n}(\cdot) \to \bar{T}_k(\cdot)$, and $\bar{Y}_j^{r_n}(\cdot) \to \bar{Y}_j(\cdot)$, where $\bar{S}_k(t) = \mu_k t$ and $\bar{V}_k(t) = m_k t$ for $t \geqslant 0$, we have

$$\begin{aligned}&\bar{V}_k^{r_n}\left(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}^{r_n}(t)\right)\\ &\quad \to \bar{T}_k(t) + m_k \bar{Z}_k(t)/B_k, \quad \text{u.o.c.}\end{aligned} \tag{5.14}$$

Hence,

$$\bar{W}_j^{r_n}(t) \to \sum_{k \in \mathscr{C}(j)} \bar{T}_k(t) + \sum_{k \in \mathscr{C}(j)} \frac{m_k}{B_k} \bar{Z}_k(t) + t - \bar{Y}_j(t), \quad \text{u.o.c.}$$

By (3.16), we have $\sum_{k \in \mathscr{C}(j)} \bar{T}_k(t) + t - \bar{Y}_j(t) = 0$. Thus,

$$\bar{W}_j^{r_n}(t) \to \sum_{k \in \mathscr{C}(j)} \frac{m_k}{B_k} \bar{Z}_k(t), \quad \text{u.o.c.},$$

proving (5.11).

By the definition of $\tau_j(t)$, we have

$$\begin{aligned}\tau_j(t) \leqslant \sum_{k \in \mathscr{C}(j)} \Big[& V_k\left(S_k(T_k(t)) + F_k(t)\right)\\ & - V_k\left(S_k(T_k(t)) + F_k(t) - 1\right)\Big].\end{aligned}$$

So

$$\begin{aligned}\bar{\tau}_j^{r_n}(t) \leqslant \sum_{k \in \mathscr{C}(j)} \Big[& \bar{V}_k^{r_n}(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}_k^{r_n}(t))\\ & - \bar{V}_k^{r_n}(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}_k^{r_n}(t) - 1)\Big].\end{aligned} \tag{5.15}$$

As in (5.14), we have

$$\begin{aligned}&\bar{V}_k^{r_n}\left(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}^{r_n}(t) - 1\right)\\ &\quad \to \bar{T}_k(t) + m_k \bar{Z}_k(t)/B_k, \quad \text{u.o.c.}\end{aligned} \tag{5.16}$$

as $n \to \infty$. Convergence (5.12) follows from (5.15), (5.14), and (5.16). □

5.3. Generalized Round Robin Policies

For a standard network, a generalized round robin (GRR) dispatch policy associated with weight parameter $\beta = (\beta_2, \ldots, \beta_K)$ is defined as follows. Here each β_k is a positive real number. Recall that $\mathscr{C}(j)$ is the set of classes at station j. We assume that the set is ordered and the order is fixed. To describe the policy, we first assume that β_k's are integers. Server j visits the ordered list of classes cyclically: once it enters class k, it serves exactly β_k jobs or exhausts the class k jobs; at the end of this period, it enters the next class on the list (or the first class if class k is the last class on the list). For a class k at the station, a cycle starting from k is defined to be the period between the time the server first enters the class and the time it reenters the class. Any class (fixed) at a station can initiate cycles. When β_k's are integers, the nominal allocation in a cycle to class k is exactly β_k, although that allocation is redistributed when the class has fewer than β_k jobs during the class k service period.

Now we let β_k's be arbitrary positive real numbers. The GRR dispatch policy works as before except that the nominal allocation to class k during a cycle needs to be adjusted. For each cycle n, let $a_k(n)$ denote the nominal allocation to class k during cycle n and $b_k(n)$ be the residual allocation to class k after cycle n. They are defined recursively as follows:

$$a_k(n+1) = \lfloor b_k(n) + \beta_k \rfloor, \tag{5.17}$$

$$b_k(n+1) = b_k(n) + \beta_k - a_k(n+1), \tag{5.18}$$

for $n = 0, 1, \ldots$, where $b_k(0) = 0$ and, as before, $\lfloor x \rfloor$ denotes the integer part of a real number x. A GRR dispatch policy is among the family of fair queueing policies widely studied in computer network literature; see, for example, Demers et al. (1989) or Parekh and Gallager (1993).

The additional standard fluid model Equation (3.22) takes the form

$$\dot{\hat{T}}_k(t) \geqslant \frac{\beta_k(m_k/B_k)}{\sum_{\ell \in \mathscr{C}(j)} \beta_\ell(m_\ell/B_\ell)}, \quad k = 1, 2, \ldots, K, \tag{5.19}$$

for each time t such that $\hat{T}_k(t)$ is differentiable and $\hat{Z}_k(t) > 0$. The intuitive explanation of (5.19) is as follows: the average cycle length is at least $\sum_{\ell \in \mathscr{C}(j)} \beta_\ell(m_\ell/B_\ell)$. When there are enough jobs in class k, the average time spent in class k during a cycle is $\beta_k m_k/B_k$. Thus, when there are enough jobs in class k, server j spends at least $\beta_k(m_k/B_k)[\sum_{\ell \in \mathscr{C}(j)} \beta_\ell(m_\ell/B_\ell)]^{-1}$ amount of effort in class k.

Now we describe the induced batch policy corresponding to the GRR dispatch policy associated with vector $\beta = (\beta_2, \ldots, \beta_K) > 0$. Here $a_k(n)$ denotes the nominal number of full class k batches to be served during cycle n. It is defined recursively through

$$a_k(n+1) = \lfloor b_k(n) + \beta_k/B_k \rfloor, \tag{5.20}$$

$$b_k(n+1) = b_k(n) + \beta_k/B_k - a_k(n+1), \tag{5.21}$$

for $n = 0, 1, \ldots$ with $b_k(0) = 0$. When server j enters class k at cycle n, it attempts to serve up to $a_k(n)$ full batches if it can. Then it moves to the next class. If β_k/B_k's are integers, the nominal number of class k batches during a cycle is β_k/B_k.

Again, for the GRR batch fluid model, it turns out that the additional fluid model Equation (3.20) takes the same form as (5.19). We offer the following proposition.

Proposition 5.7. *For each fluid limit $\bar{\mathbb{X}}$ of the batch-processing network operating under the induced GRR batch policy, we have*

$$\dot{\bar{T}}_k(t) \geqslant \frac{(\beta_k/B_k)m_k}{\sum_{\ell\in\mathscr{C}(j)}(\beta_\ell/B_\ell)m_\ell}, \tag{5.22}$$

for all time t such that $\bar{T}_k(t)$ is differentiable and $\bar{Z}_k(t) > 0$, $k = 1, 2, \ldots, K$.

The proof is provided at the end of this section.

Proposition 5.8. *Any GRR dispatch policy is a normal policy.*

Proof. Let $\bar{\mathbb{X}}$ be a fluid solution to the batch fluid model under the induced GRR batch policy. Let $\hat{\mathbb{X}}$ be a fluid solution constructed from $\bar{\mathbb{X}}$ by (4.1)–(4.6). Then at any time t such that $\hat{Z}_k(t) > 0$ and $\dot{\hat{Z}}_k(t)$ exists, we have $\bar{Z}_k(t) > 0$ and $\dot{\bar{Z}}_k(t)$ exists. Thus, by Proposition 4.3 and (5.22), we have

$$\dot{\hat{T}}_k(t) = \dot{\bar{T}}_k(t) \geqslant \frac{(\beta_k/B_k)m_k}{\sum_{\ell\in\mathscr{C}(j)}(\beta_\ell/B_\ell)m_\ell}.$$

Thus, $\hat{\mathbb{X}}$ satisfies (5.19) and, hence, is a standard fluid model solution. □

Corollary 5.9. *Let $\beta = (\beta_2, \ldots, \beta_K)$ be a vector of positive real numbers. Assume that for each class k,*

$$\frac{(\beta_k/B_k)m_k}{\sum_{l\in\mathscr{C}(j)}(\beta_\ell/B_\ell)m_\ell} \geqslant \lambda_k m_k/B_k. \tag{5.23}$$

Then the batch-processing network operating under the induced GRR batch policy with weight β is rate stable.

Proof. Let $\hat{\mathbb{X}}$ be a standard fluid model solution with $\hat{Z}(0) = 0$. Under conditions (5.19) and (5.23), $\hat{D}_k(t) \geqslant \lambda_k$ for time t such that $\hat{Z}_k(t) > 0$ and $\hat{\mathbb{X}}$ is differential at t. It follows the proof of Theorem 4 of Bramson (1998) that $\hat{Z}(t) = 0$ for $t \geqslant 0$. Thus, the standard GRR fluid model is weakly stable. Since any GRR dispatch policy is normal, the corollary follows from Theorem 2.1. □

Notice that condition (5.23) is equivalent to

$$\lambda_k \left(\sum_{\ell\in\mathscr{C}(j)} (\beta_\ell/B_\ell)m_\ell \right) < \beta_k, \quad k = 1, \ldots, K.$$

The latter form of the condition has the following intuitive interpretation: the average number of job arrivals to class k during a cycle is less than the number of class k jobs that can be served during a cycle. When the usual traffic condition is satisfied, one can find a weight parameter β that satisfies the condition.

Proof of Proposition 5.7. Let $\bar{\mathbb{X}}$ be a fluid limit of the batch-processing network with the corresponding sequence $r_n \to \infty$ such that $\bar{\mathbb{X}}(\cdot) = \lim_{n\to\infty} \bar{\mathbb{X}}^{r_n}(\cdot)$. We would like to show that (5.22) holds for $\bar{\mathbb{X}}$.

Let $u > 0$ be a time such that $\bar{T}(\cdot)$ is differential and $\bar{Z}_k(u) > 0$. By the continuity of $\bar{Z}$, there exists a $\delta > 0$ such that $\bar{Z}_k(s) > 0$ for $s \in (u-\delta, u+\delta)$. It suffices to show that

$$\frac{\bar{T}_k(t) - \bar{T}_k(s)}{t-s} \geqslant \frac{(\beta_k/B_k)m_k}{\sum_{l\in\mathscr{C}(j)}(\beta_\ell/B_\ell)m_\ell},$$

for any $u - \delta < s < t < u + \delta$. Since $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ as $n \to \infty$, it is enough to show that

$$\lim_{n\to\infty} \frac{T_k(r_n t) - T_k(r_n s)}{r_n(t-s)} \geqslant \frac{(\beta_k/B_k)m_k}{\sum_{l\in\mathscr{C}(j)}(\beta_\ell/B_\ell)m_\ell}. \tag{5.24}$$

To study the limit in (5.24), we focus the batch-processing network in the time interval $[r_n s, r_n t]$ for large n. Let C_n be the number of cycles that are initiated and completed in the time interval. Note that time $r_n s$ may be in the middle of a cycle that was initiated before time $r_n s$, and time $r_n t$ may be in the middle of a cycle that ends after $r_n t$. Following the same argument as in Proposition 5.1, one can choose n large enough and δ small enough such that

$$Z_k(t') > \beta_k + B_k \quad \text{for } t' \in (r_n s, r_n t). \tag{5.25}$$

Thus, class k always forms full batches in any one of the C_n cycles in $(r_n s, r_n t)$.

Define $G_\ell(t')$ to be the number of class ℓ batches completed by time t'. Then $V_\ell(G_\ell(t'))$ is the time spent by server $j = \sigma(\ell)$ to complete these batches. We then have

$$T_k(r_n t) - T_k(r_n s) \geqslant V_k(G_k(r_n t)) - V_k(G_k(r_n s) + 1).$$

The difference between the two sides is due to the remaining processing time of batch $G_k(r_n s)$ and the time already spent on batch $G_k(r_n t) + 1$. By (5.25), there are at least $\lfloor C_n \beta_k/B_k \rfloor$ class k batches that have been initiated and completed in $(r_n s, r_n t)$. Thus,

$$\begin{aligned} T_k(r_n t) - T_k(r_n s) \geqslant{}& V_k(G_k(r_n s) + 1 + \lfloor C_n \beta/B_k \rfloor) \\ &- V_k(G_k(r_n s) + 1). \end{aligned}$$

Similarly, since server $j = \sigma(k)$ has been busy in $(r_n s, r_n t)$, we have

$$r_n t - r_n s \leqslant \sum_{\ell\in\mathscr{C}(j)} V_\ell(G_\ell(r_n t) + 1) - V_\ell(G_\ell(r_n s)).$$

Because there are at most $C_n + 2$ cycles that have been initiated or completed in $(r_n s, r_n t)$ (C_n full cycles and 2 partial cycles), class ℓ has at most $\lfloor (C_n + 2)\beta_\ell/B_\ell \rfloor$ batches

that have been served (completed or initiated) in $(r_n s, r_n t)$, $G_\ell(r_n t) - G_\ell(r_n s) \leqslant \lfloor (C_n+2)\beta_\ell/B_\ell \rfloor$. Hence,

$$r_n t - r_n s \leqslant \sum_{\ell \in \mathscr{C}(j)} V_\ell\left(G_\ell(r_n s) + \lfloor (C_n+2)\beta_\ell/B_\ell \rfloor\right) - V_\ell(G_\ell(r_n s)).$$

Therefore, we have the following inequality:

$$\frac{T_k(r_n t) - T_k(r_n s)}{r_n(t-s)} \geqslant \frac{C_n^{-1}[V_k(G_k(r_n s)+1+\lfloor C_n\beta/B_k \rfloor) - V_k(G_k(r_n s)+1)]}{\sum_{\ell \in \mathscr{C}(j)} C_n^{-1}[V_\ell(G_\ell(r_n s)+\lfloor (C_n+2)\beta_\ell/B_\ell \rfloor) - V_\ell(G_\ell(r_n s))]}. \tag{5.26}$$

We now claim that

$$\lim_{n\to\infty} C_n^{-1}\Big[V_k(G_k(r_n s)+1+\lfloor C_n\beta/B_k \rfloor) - V_k(G_k(r_n s)+1)\Big] = \beta_k m_k/B_k. \tag{5.27}$$

The claim follows from assumption (2.1) and an extension of the strong-law-of-large-numbers (see, for example, Lemma 5.2.1 of Jennings 2000a), provided that

$$\limsup_{n\to\infty} G_k(r_n s)/C_n < \infty. \tag{5.28}$$

Since $G_k(r_n s) \leqslant S_k(r_n s)$ and $\lim_{n\to\infty} S_k(r_n s)/r_n \to \mu_k$, (5.28) follows from

$$\liminf_{n\to\infty} \frac{C_n}{r_n} > 0, \tag{5.29}$$

which means that C_n increases at least at the same rate as r_n.

To prove (5.29), for each class $\ell \in \mathscr{C}(j)$, because server j can visit class ℓ at most C_n+2 times in $(r_n s, r_n t)$ and each time the server can work at most $(\beta_\ell + B_\ell)$ class ℓ jobs, we have

$$\begin{aligned}\liminf_{n\to\infty} \frac{(C_n+2)(\beta_\ell+B_\ell)}{r_n} &\geqslant \lim_{n\to\infty} \frac{D_\ell(r_n t) - D_\ell(r_n s)}{r_n} \\ &= \frac{\bar{D}_k(t) - \bar{D}_k(s)}{t-s} \\ &= \mu_k \frac{\bar{T}_k(t) - \bar{T}_k(s)}{t-s},\end{aligned}$$

where the last equality follows from (3.19). Moving μ_ℓ to the other side and summing up for $\ell \in \mathscr{C}(j)$, we have

$$\left(\sum_{\ell \in \mathscr{C}(j)} m_\ell(\beta_\ell + B_\ell)\right) \liminf_{n\to\infty} C_n/r_n = \sum_{\ell \in \mathscr{C}(j)} (\bar{T}_\ell(t) - \bar{T}_\ell(s)) = t-s,$$

where the last equality follows from (3.16) and (3.17). Hence (5.29) is true, and thus (5.27) holds.

Similarly, for each class $\ell \in \mathscr{C}(j)$, we can prove

$$\lim_{n\to\infty} C_n^{-1}[V_\ell(G_\ell(r_n s) + \lfloor (C_n+2)\beta_\ell/B_\ell \rfloor) - V_\ell(G_\ell(r_n s))] = \beta_\ell m_\ell/B_\ell. \tag{5.30}$$

Inequality (5.24), and hence the proposition, follows from (5.27), (5.30), and (5.26). □

6. EXTENSIONS

Throughput is an important performance measure for batch-processing networks. But it is a crude one. One would like to further differentiate policies that are throughput optimal. This differentiation is based on some secondary performance measure, like minimizing the long-run average number of jobs in a system. Unfortunately, the mode of analysis in this paper offers little insight towards the objective. In the case that the nominal utilization for some station is well below one and the maximum batch sizes at the station are large, full batch policies, though stable, may not be desirable for the secondary performance measure. Suppose that one can choose b_k's with $1 \leqslant b_k \leqslant B_k$ such that

$$\sum_{k \in \mathscr{C}(j)} \lambda_k(m_k/b_k) \leqslant 1, \quad j = 1, \ldots, J.$$

One can relax full batch policies by allowing any class k with at least b_k jobs to be treated as nonempty. Whenever a server selects the next class to form a batch, all "nonempty" classes are eligible to be chosen. Once a class k is chosen, the server loads up to B_k jobs for the batch. The size of the batch may be smaller than B_k, but it is at least b_k.

Note that while B_k represents a physical restriction from a piece of equipment, b_k comes from a management decision. Once $b = (b_1, \ldots, b_K)'$ is chosen and fixed, an analogous theory based on fluid models can be developed to prove the stability of the relaxed batch policies. Choosing b to minimize the secondary performance measure is important, but is beyond the scope of this paper.

ACKNOWLEDGMENTS

Research supported in part by NSF grants DMI-9457336 and DMI-9813345, by a Chinese NSF grant, and by TLI-AP, a partnership between National University of Singapore and Georgia Institute of Technology.

REFERENCES

Bramson, M. 1994. Instability of FIFO queueing networks. *Ann. Appl. Probab.* **4** 414–431.

——. 1996a. Convergence to equilibria for fluid models of FIFO queueing networks. *Queueing Systems: Theory Appl.* **22** 5–45.

——. 1996b. Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks. *Queueing Systems: Theory Appl.* **23** 1–26.

——. 1998. Stability of two families of queueing networks and a discussion of fluid limits. *Queueing Systems: Theory Appl.* **28** 7–31.

Chen, H. 1995. Fluid approximations and stability of multiclass queueing networks I: Work-conserving disciplines. *Ann. Appl. Probab.* **5** 637–665.

——, H. Zhang. 1997a. Diffusion approximations for reentrant lines with a first-buffer-first-served priority discipline. *Queueing Systems: Theory Appl.* **23** 177–195.

——, ——. 1997b. Stability of multiclass queueing networks under FIFO service discipline. *Math. Oper. Res.* **22** 691–725.

——, ——. 2000. Stability of multiclass queueing networks under priority service disciplines. *Oper. Res.* **48** 26–37.

Dai, J. G. 1995. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Ann. Appl. Probab.* **5** 49–77.

——. 1999. *Stability of Fluid and Stochastic Processing Networks.* MaPhySto Miscellanea Publication, No. 9 ⟨http://www.maphysto.dk/⟩.

——, S. P. Meyn. 1995. Stability and convergence of moments for multiclass queueing networks via fluid limit models. *IEEE Trans. Automatic Control* **40** 1889–1904.

——, J. VandeVate. 2000. The stability of two-station multi-type fluid networks. *Oper. Res.* **48** 721–744.

——, G. Weiss. 1996. Stability and instability of fluid models for re-entrant lines. *Math. Oper. Res.* **21** 115–134.

Demers, A., S. Keshav, S. Shenker. 1989. Analysis and simulation of a fair queueing algorithm. *Proc. Sigcomm.* **19**(4) 1–12.

El-Taha, M., S. Stidham Jr. 1999. *Sample-Path Analysis of Queueing Systems*. Kluwer, Norwell, MA.

Harrison, J. M. 1998. Brownian models of queueing networks with heterogeneous customer populations. W. Fleming, P. L. Lions, eds. *Stochastic Differential Systems, Stochastic Control Theory and Their Applications*, Vol. 10. *The IMA Volumes in Mathematics and Its Applications*. Springer, New York, 147–186.

——, V. Nguyen. 1993. Brownian models of multiclass queueing networks: Current status and open problems. *Queueing Systems: Theory Appl.* **13** 5–40.

Hasenbein, J. J. 1997. Necessary conditions for global stability of multiclass queueing networks. *Oper. Res. Lett.* **21** 87–94.

Jennings, O. B. 2000a. Multiclass queueing networks with setup delays: Stability analysis and heavy traffic approximation. Ph.D. thesis, School of Industrial and System Engineering, Georgia Institute of Technology, Atlanta, GA.

——. 2000b. On the stability of multiclass queueing networks with setups. Preprint.

Kumar, P. R. 1993. Re-entrant lines. *Queueing Systems: Theory Appl.* **13** 87–110.

——, T. I. Seidman. 1990. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Trans. Automatic Control* **AC-35** 289–298.

Kumar, S., P. R. Kumar. 1996. Fluctuation smoothing policies are stable for stochastic reentrant lines. *Discrete Event Dynamical Systems* **6** 361–370.

——, H. Zhang. 2000. Stability of reentrant lines with batch servers. Technical report, Graduate School of Business, Stanford University, Stanford, CA.

Lu, S. H., P. R. Kumar. 1991. Distributed scheduling based on due dates and buffer priorities. *IEEE Trans. Automatic Control* **36** 1406–1416.

Maglaras, C., S. Kumar. 1999. Capacity realization in stochastic batch-processing networks using discrete review policies. Technical report, Graduate School of Business, Stanford University, Stanford, CA.

Parekh, A. K., R. G. Gallager. 1993. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Networking* **1** 344–357.

Rybko, A. N., A. L. Stolyar. 1992. Ergodicity of stochastic processes describing the operation of open queueing networks. *Problems Information Transmission* **28** 199–220.

Seidman, T. I. 1994. 'First come, first served' can be unstable! *IEEE Trans. Automatic Control* **39** 2166–2171.

Stolyar, A. L. 1995. On the stability of multiclass queueing networks: a relaxed sufficient condition via limiting fluid processes. *Markov Processes Related Fields* **1** 491–512.