Dynamic Scheduling of Multiclass Queueing Networks

A Thesis
Presented to
The Academic Faculty

by

Caiwei Li

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in Industrial Engineering

Georgia Institute of Technology
November 2001

Copyright © 2001 by Caiwei Li

Dynamic Scheduling of Multiclass Queueing Networks

Approved:

Prof. Jiangang Dai, Chairman

Prof. Leon McGinnis

Prof. Richard Serfozo

Prof. John Vande Vate

Prof. Yang Wang

Date Approved _____

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Jim Dai for his direction, support and feedback. His genius, patience and deep insights make it a pleasure to work with him. I also acknowledge the help and support of the other members of my committee, Professors Leon McGinnis, Richard Serfozo, John Vande Vate and Yang Wang. I would also thank the Virtual Factory Lab for providing computing resources during my four years research. Particularly, I thank Dr. Douglas Bodner for his support. My appreciation goes out to the entire school of ISyE at Georgia Tech, students and faculty, for their support and help. I would especially like to thank Ki-Seok Choi for his willingness to help me. In particular, I owe much to Zheng Wang who had the substantial tasks of proof-reading a draft of this thesis. On a more personal level, I would like to thank my friends, Jianbin Dai and Sheng Liu, for their help during my study at Georgia Tech.

I would like to thank the National Science Foundation, which has supported my research through grants DMI-9457336 and DMI-9813345. I also thank Brooks Automations Inc., AutoSimulations division for donating AutoSched AP software and providing technical support. I can hardly imagine how this research could be done without the AutoSched AP software.

Finally, I thank my family for their love and support throughout. Particularly, I thank my wife Miao Liu for her continuous support and encouragement.

Contents

List of Tables

List of Figures

Summary

This thesis presents several dispatch policies for multi-class queueing networks and studies their stability properties. The discrete proportional processor sharing (DPPS) dispatch policies are shown to be stable as long as each server's traffic intensity is less than one. Any policy can be embedded into a DPPS policy and the new policy is always stable. Largest weighted upstream unbalance (LWUU) policies and largest weighted total unbalance (LWTU) policies are two families of policies which use downstream or upstream information. For more restricted network models, we show that both LWUU and LWTU policies are stable.

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 thesis 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.

A series of simulation studies is conducted in a three-product-five-station network. The simulation results show that DPPS dispatch policies and their induced batch policies have good performance.

Chapter 1

Introduction

1.1 Motivation of the Model

Semiconductor wafer fabrication lines are among the most difficult systems for planning and scheduling. The major difficulty stems from the reentrant process flows of such lines. Typically converting into semiconductor product requires a wafer hundreds of processing steps. In a wafer fab, there are many processing stations. Each station is equipped with one or more machines. Very few machines are dedicated to a particular processing step. Instead, many machines can carry out several different but similar processing steps by simply changing configurations or tools. Usually, the number of processing steps that a station carries out is greater than the total number of machines at that station. So one can not simply dedicate one machine to each processing step. A job consisting of a cassette of wafers needs to visit some stations several times and thus jobs at different steps contend with each other for machines at such a station. Because jobs visit some machines multiple times at different processing steps, the process flows have a reentrant structure.

When a machine completes a job, it has to decide which job to process next. A policy specifying such decisions is called a dispatch policy or service policy or sequencing policy. Since a wafer fab is an extremely complex system, it is unlikely

one can find an optimal dispatch policy. Often, simple heuristic policies such as first-in-first-out (FIFO) are used. Dispatch policies are important to good performance in a fab. They impact not only on the cycle times, but also the throughput. Poor dispatch policies can lead to the loss of throughput, longer cycle time and higher work-in-process (WIP). In a wafer fab, WIP never grows to infinity since managers will slow down the release of new jobs if WIP is too high. A good dispatch policy guarantees maximum throughput. Moreover, a good policy also ensures better performance with respect to secondary performance measures such as average cycle time.

In this thesis, we discuss several dispatch policies, including Discrete Proportional Processor Sharing (DPPS), Largest Weighted Up Stream Unbalanced (LWUU), and Largest Weighted Total Unbalanced (LWTU). A DPPS policy is a discrete version of an ideal head-of-line proportional-processor-sharing (HLPPS) policy. The HLPPS policy studied in Bramson [3] can not be used for practical purpose because it requires each server to be able to split effort to many jobs simultaneously. Simultaneous splitting is impossible for almost all manufacturing lines or even computer and telecommunication systems. Our DPPS policies can be implemented in real systems. Also, DPPS policies have the same fluid limit model as the corresponding HLPPS policy. (See Section 4.4 for a discussion of fluid limits.) Moreover, each DPPS policy can be treated as more than merely a dispatch policy. In fact, it can be used as a scheme to stabilize other policies. In other words, any non-idling policy can be combined with a DPPS policy so that the resulting policy is stable for any multi-class queueing network as long as the traffic intensities at each stations is less than one. LWUU and LWTU are special policies in a family of more general policies, called MIVSRP. Li, Tang and Collins [34] demonstrated through simulation that MIVSRP policies give

smaller cycle times than FIFO. In this thesis, we demonstrate the stability properties of DPPS, LWUU and LWTU policies. Moreover, we demonstrate through simulation that those policies give better performance than FIFO.

A machine may need additional operations such as changing tools before it switches from one processing step to another. Although an operator may do some of these operations while the machine is processing, for others an operation must wait until the machine finishes a job, which causes an additional delay. This delay is called *setup* delay, and the amount of delay is called *setup time*. In some cases, the setup time can be as much as ten times of the processing time. When a long setup occurs, one also needs to decide whether it should change from one processing step to another. When such a decision is involved, the policy that a machine follows is called a setup policy. Jennings [21] studied the stability of setup policies and gave a general framework for turning a good dispatch policy into a good setup policy.

Machines may involve batch operations. One good example is a furnace which can often process up to a dozen jobs at a time. When there is a batch operation, one may also need to decide whether it should wait for full batch or not. In that case, the policy is called a batch policy. In this thesis, we provide a general framework for converting a dispatch policy into a batch policy. We call such a batch policy an induced batch policy. Our framework guarantees that the induced batch policy is throughput optimal if the corresponding dispatch policy is. While we are not able to prove similar result for mean cycle times, we can demonstrate through simulation that batch policies induced from good dispatch policies also give good secondary performance characteristics such as short mean cycle times.

1.2 Literature Review

Recently, stability of multiclass queuing networks has received a lot of attention in the research community. This is due to many examples of queuing networks that are unstable under the usual traffic intensity conditions. The first example was found by Kumar and Seidman [24]. Later Lu and Kumar [27] gave an example that is unstable under the Static Buffer Priority (SBP) policy. Rybko and Stolya [32] studied the stochastic version of the Lu-Kumar network, and found that SBP policy used in Lu and Kumar [27] is also unstable in a stochastic setting. Bramson [1] presented an exponential queuing network that is unstable under the FIFO policy when the usual traffic condition is satisfied. At the same time, Seidman [33] independently found an unstable FIFO deterministic queuing network. Perhaps, because FIFO is such a natural and fair policy, and widely used in practice, these instable examples have motivated more researchers to work in the field. Dai [10] provided a connection between the stability of a fluid model and the stability of the corresponding stochastic queueing network in Dai [10]. See also Stolyar [35]. Since then, many policies have been proven to be stable. Dai and Weiss [14] and Kumar and Kumar [25] proved Last-buffer-first-server (LBFS) and first-buffer-first-serve (FBFS) are stable in single product reentrant line. Bramson [2] proved FIFO policy is stable for Kelly type queuing networks. Jennings [21] proved that Generalized Round Robin (GRR) is stable for all queuing networks even with setup delays when its parameters are chosen properly. Bramson [3] proved HLPPS is stable. In order to guarantee stability, GRR needs to know the arrival rates of the system. In a real factory, because arrival rates fluctuate from week to week, they are difficult to estimate. HLPPS can guarantee optimal

throughput without the information of arrival rates as long as the traffic intensity is less than one at each station. However, the HLPPS discussed in Bramson [3] can not be used for practical purposes because it requires servers to split their effort among multiple jobs simultaneously. More recently, Bramson [5] showed that Earliest Due Date (EDD) is stable for all queueing networks. Maglaras's [28] fluid tracking policies are stable for all networks. Maglaras [29] showed that Harrison's discrete review policies are stable for all networks.

Most of the stability analysis in the literature has been limited to standard processing networks, also called *multiclass queueing networks*, as advanced by Harrison [18]. Two exceptions are Maglaras and Kumar [30] and Kumar and Zhang [26], which studied batch processing networks. In [30], a family of discrete review batch policies was shown to maximize the throughput. In [26], a family of fluctuation-smoothing batch policies was shown to maximize the throughput in special networks,which Kumar [23] calls reentrant lines.

1.3 Contributions

The specific contributions of this thesis begin from Part I with proposing or improving several family policies. The first family of policies, the discrete proportional processor sharing (DPPS) policies, are related to the head-of-line proportional processor sharing (HLPPS) policy proposed by Bramson [3]. In Bramson [3], HLPPS is proved to be stable for all open multiclass queueing networks in which the traffic intensities of all stations are less than one. However, there is one assumption of HLPPS policy. It assumes that each server can serve more than one job at a time. In other words, each

server can split its service effort infinitely among several jobs. This assumption makes the HLPPS policy impossible to implemente at least in real wafer fabs. DPPS policies relax this unrealistic assumption and thus may be implemented in real systems. Moreover, we prove that the fluid limits of queueing networks under DPPS and those of queueing networks under HLPPS are identical. A DPPS policy is a realistic policy in the sense that it can be implemented in real systems. In addition, it only makes high level decisions and leaves detailed sequencing decisions open. More concretely, during any decision period, a DPPS policy only specifies the number of jobs of each class to be processed without determining the processing sequence of those jobs. One can use any other policy or optimization procedure to further specify the sequence. For a wafer fab the arrival rates fluctuate from week to week making any policy that requires knowledge of the arrival rates undesirable. A DPPS policy does not need information about arrival rates, which makes it a practical policy.

The second family of policies is the largest weighted upstream unbalance (LWUU) policies. LWUU policies assign a target queue length to each buffer and dynamically change each buffer's priority according to upstream buffers' imbalance information. The server always gives higher priority to buffers with higher weighted upstream imbalance. For each buffer (p, k), the upstream imbalance is defined as

$$\beta_p \sum_{l \leq k} (Z_{p,l}(t) - \xi_{p,l}),$$

where $Z_{p,l}(t)$ is the number of jobs in class (p, l) at time t, $\xi_{p,l}$ is the target queue length for buffer (p, l), and β_p is some positive constant.

The third family of policies, the largest weighted total unbalance (LWTU) policies, use both upstream and downstream information. Similar to LWUU policies, LWTU

policies also assign a target queue length to each buffer and dynamically change each buffer's priority according to both upstream and downstream buffers' imbalance information. The server always gives higher priority to larger weighted total unbalance defined by:

$$\beta_p \sum_{l \le k} (Z_{p,l}(t) - \xi_{p,l}) - \beta_p \sum_{l > k} (Z_{p,l}(t) - \xi_{p,l}).$$

For second and third family of policies, we show that for deterministic routing multi product queueing networks, all these policies are stable as long as the long term traffic intensity at each station is less than one.

Batch processing is a very important feature of semiconductor reentrant lines. One example is a furnace. A furnace usually can processes up to a dozen jobs at one time. The processing time can typically be as long as 8 hours which is 100 times longer than the processing times in other areas. Because of its long processing time, a furnace has a very significant impact on the performance of the entire system. Similar to the stations in other areas, several classes of jobs are waiting to enter a furnace. The batch policy specifies which class of jobs to process next when the furnace finishes a batch. Since the capacity of a furnace depends on the batch size, one has to be careful when deciding to process a batch whose size is less than the maximal batch size. On the other hand, which class to process is still very important because this will affect the overall flow through the entire system.

There are a number of efficient dispatch policies for the standard processing networks in which the batch size is restricted to one. To take advantage of these dispatch policies when batch sizes are allowed to exceed one, we propose an algorithm to convert a dispatch policy to a batch policy. The basic idea is that a station always chooses those classes with enough jobs to form a full batch. Hence the capacity of

a station is protected. Among those classes, use the dispatch policy to determine which class to work on. Since we know that the original dispatch policy has good performance, intuitively we would expect that the converted batch policy should have good performance as well. In fact, we show that if the original dispatch policy is a "normal" policy and can achieve maximal throughput, the converted batch policy can also achieve maximal throughput. The definition of normal policy is through fluid models. We demonstrate that static buffer priority (SBP), FIFO, and Generalized Round Robin (GRR) are all normal policies.

1.4 Outline of the Thesis

The remaining part of this thesis is organized into three parts. Part I including Chapters 2, 3, 4, and 5 addresses the dispatch policies. In Chapter 2, we introduce standard multiclass queueing networks. In Chapter 3, we focus on three families of dispatch policies: the discrete proportional processor sharing (DPPS) policies, the largest weighted upstream unbalance first (LWUU) policies, and the largest weighted total unbalance first (LWTU) policies. In Chapter 4, we introduce an important analysis tool—-fluid models, as well as other related conceptions, such as rate stability and fluid limits. In chapter 5, using the fluid model approach and assumption that the traffic intensity at each stations is less than one, we show that (1) any DPPS policy is rate stable for any standard multiclass queueing network; (2) LWUU and LWTU polices are rate stable for any standard multiclass queueing network with deterministic routing. Part II includes Chapters 5, 7, 8, and 9. In part II, we will focus on batch processing networks and their relation with standard processing networks. In

Chapter 6, 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 this part. In Chapter 7, 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 we discuss fluid limits that are used to justify the fluid equations defining a fluid model. In Chapter 8, we study the relationship between the fluid models of processing networks. Using this relationship, we then define *normal dispatch policies* in a standard network, a key notion used in the statement of our main theorem, which saying the the induced batch policy can preserve stability properties of the original dispatch policy. Finally, we present examples of normal dispatch policies in Chapter 9. These include the static buffer priority, first-in–first-out, and generalized round robin policies. In Part III, we conduct several simulation studies. First we introduce a 3-product-5-station network which serves as our test bed. We conduct three simulation studies. First we simulate the network under several dispatch policies, assuming no batch operations or setup delays. Second, we add batch operations and simulate the network under several batch policies. Finally, we add both batch operations and setup delays to the network and simulate the network under various batch and setup policies.

Part I

Standard Multiclass Queueing Networks

Chapter 2

Stochastic Processing Networks

In this chapter, we describe the model that is the subject of this thesis. In Section 2.1, we introduce the multiclass queueing network model. Section 2.2 discusses three primitive processes which are basic mathematical elements of our model.

2.1 Multiclass Queueing Network Models

The network under study has J single-server stations and K job classes. Stations are labeled by $j = 1, ..., J$ and classes by $k = 1, ..., K$. Class k jobs are served at a unique station $\sigma(k)$. Each station may serve more than one class and has an unlimited buffer capacity for each job class. Jobs arrive at the network from outside, and change classes as they move through it. When a job finishes its processing, it is routed to the next class or if it has completed processing of all steps, leaves the network. If a job is routed to another class, it enters the buffer of that class and waits at the end of line. Each job eventually leaves the network. The ordered sequence of classes that a job visits in the network is called a route. 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 several times. One can also extend reentrant lines to multiple product reentrant lines. In this case, there are P products. All jobs of a particular product p

follow a deterministic route with K_p stages. Again jobs may visit some stations more than once. Different products can have different routes.

2.2 Primitive Processes

We let $\mathcal{C}(j)$ denote the set of classes that are served by station j. When j and k appear together, we imply that $j = \sigma(k)$. For each class k, there are three cumulative processes $E_k = \{E_k(t), t \geq 0\}$, the external arrival process; $V_k = \{V_k(n) : n = 1, 2, ...\}$, the service time process; and $\Phi^k = \{\Phi^k(n) : n = 1, 2, ...\}$, the job routing process. For each time $t \geq 0$, $E_k(t)$ counts the number of external arrivals to class k in $[0, t]$ and $V_k(n)$ is the total service time requirement for the first n jobs 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 service at class k. By convention, we assume

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

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

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

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

the primitive processes, namely, with probability one,

$$\lim_{t\to\infty} \frac{E_k(t)}{t} = \alpha_k, \quad \lim_{t\to\infty} \frac{V_k(t)}{t} = m_k, \quad \text{and} \quad \lim_{t\to\infty} \Phi_\ell^k(t)/t = P_{k\ell}, \tag{2.2.1}$$

where $k,\ \ell = 1,\ldots,K$. The parameter (α, m, P) with $\alpha = (\alpha_1,\ldots,\alpha_K)'$, $m = (m_1,\ldots,m_K)'$ and $P = (P_{k\ell})$ has the following natural interpretation: 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 jobs. For classes k and ℓ, $P_{k\ell}$ is the long-run fraction of class k jobs that expect to move to class ℓ. The parameter $P_{k\ell}$ 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

$$Q = I + P' + (P')^2 + ...$$

is finite, which is equivalent to the assumption that $(I - P')$ is invertible and $Q = (I - P')^{-1}$.

Whenever a server is ready to load a job, it must decide which job to serve next. A policy for making this decision is called a *dispatch policy*. We assume that once a service is started, it cannot be preempted. A dispatch policy is said to be non-idling if a server remains active whenever there are jobs at its station.

Chapter 3

Dispatch Policies

Consider a standard queueing network without batch operations and setup delays. When a server completes processing of one operation it must decide which class to work on next. When there is no setup delay and batch operation involved, we call this decision a dispatch policy. Many dispatch policies have been proposed in the literature, including first-in-first-out (FIFO), last-buffer-first-served (LBFS), and first-buffer-first-served (FBFS). We restrict our dispatch policies to non-idling head-of-line (HL) dispatch policies. Under an HL dispatch policy, each class has at most one job (the leading job) that is ready to be served by a server. Jobs within a class are served in FIFO order. In this section, we will discuss several families of dispatch policies, including the Discrete Proportional Processor Sharing (DPPS) policies, the Largest Weighted Up Stream Unbalanced (LWUU) policies and the Largest Weighted Total Unbalanced (LWTU) policies. All of these policies are HL dispatch policies.

3.1 Discrete Proportional Processor Sharing

In this section, we introduce Discrete Proportional Processor Sharing (DPPS) dispatch policies. Since DPPS policies are closely related to the head-of-line proportional-processor-sharing (HLPPS) dispatch policy studied in Bramson [3], we first review

the HLPPS policy.

When a stochastic network operates under the HLPPS policy, jobs in each buffer are ordered according to the jobs' arrival time to the buffer. At each time t, the leading job from each nonempty buffer k receives a portion of service from server $i = \sigma(k)$. Thus, server i simultaneously works on all the nonempty buffers at the station. The amount of service effort received by the leading job in a nonempty class is proportional to the number of jobs in that class. Denoting by $T_k(t)$ the cumulative amount of time that server $i = \sigma(k)$ has spent on class k jobs in $[0, t]$, one has

$$\dot{T}_k(t) = \frac{Z_k(t)}{U_i(t)} \quad \text{for } t \geq 0, \tag{3.1.1}$$

where $\dot{T}_k(t)$ is the derivative of T_k at time t, $Z_k(t)$ is the number of jobs in buffer k at time t, including the one being served, and $U_i(t) = \sum_{k \in \mathcal{C}(i)} Z_k(t)$ is the total number of jobs at station i. Each time a service is completed at a station, the server adjusts its service effort allocation following the new queue lengths at the station.

The HLPPS dispatch policy assumes that at any given time each server is able to split its service effort among multiple jobs in an arbitrary fashion. Such an assumption does not hold for most real world systems. Our DPPS policies, introduced in the next paragraph, try to mimic the HLPPS policy. The important difference is that, under a DPPS policy, each server works on only one job at a time and preemption of jobs is not allowed. Thus, under a DPPS policy condition (3.1.1) does not hold at each time $t \geq 0$. We will show that (3.1.1) holds for each "fluid limit" under DPPS policy.

When the network operates under a DPPS policy, each server makes its service allocation decision independent of other servers. Server i makes decisions at a sequence of decision times $t_0, t_1, \ldots$. For each integer $n = 0, 1, \ldots$, t_n is called the nth decision

epoch, and $[t_n, t_{n+1})$ is called the nth cycle. At the beginning of the nth cycle t_n, server i decides how many jobs to serve for each class during the cycle based on the queue lengths at its station. The cycle ends when all the planned jobs are served, and the next cycle starts.

Fix a station i. The length of the nth cycle at the station depends on a parameter L_n which is called *the nominal length* of the nth cycle. (In general, L_n depends on station i. For convenience, we drop the index i from the symbol L_n.)

Suppose that the nominal length L_n has been chosen at t_n,the beginning of the nth cycle. This length is to be split among all the classes at the station. The amount that class k receives is proportional to $Z_k(t_n)/U_i(t_n)$. (The ratio 0/0 is interpreted as zero.) Thus, the nominal service time that class k receives during the cycle is $(Z_k(t_n)/U_i(t_n))L_n$. Since the service speed for class k jobs is μ_k, we plan to serve

$$(Z_k(t_n)/U_i(t_n))L_n\mu_k \tag{3.1.2}$$

class k jobs during the cycle.

There are two potential problems in our schedule. First, there may be fewer jobs available at t_n in a class than planned by our schedule. Secondly, the number in (3.1.2) may not be an integer. To resolve the first problem, we serve as many as planned class k jobs that are available during the cycle. Some of these jobs may arrive at the station *after* t_n. For the second problem, we simply truncate the number in (3.1.2) to an integer. But the *residual number* $r_k(n)$ is retained for the next cycle when the ending buffer level is positive.

To summarize, for a given station i and nominal length L_n, define for each cycle

n and each class k,

$$q_k(n) = r_k(n-1) + \frac{Z_k(t_n)}{U_i(t_n)} L_n \mu_k, \tag{3.1.3}$$

$$\beta_k(n) = \lfloor q_k(n) \rfloor, \tag{3.1.4}$$

$$r_k(n) = [q_k(n) - \beta_k(n)] 1_{\{Z_k(t_{n+1})>0\}}, \tag{3.1.5}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, and $r_k(0) = 0$. We call $q_k(n)$ the *quota* for class k in cycle n, and $r_k(n)$ the *residual quota* for class k after cycle n. Note that the residual quota is saved to the next period only when the queue length of that class is positive at the end of the cycle (or the beginning of next cycle). The integer $\beta_k(n)$ is the number of class k jobs that should be processed in cycle n. $\beta_k(n)$ may not be used up during nth cycle. We denote this part as $p_k(n)$ and it is lost after the cycle n.

The choice of L_n can be quite flexible. It can be dynamic or static. In the former case, we require L_n to be bounded above by a constant. This ensures that even if some buffers have a huge number of jobs during a cycle, the cycle will end reasonably soon. The DPPS policy does not specify in which order the planned jobs are served during a cycle. Actually, within a cycle, one can use any other dispatch policy to determine the order for serving the jobs. More concretely, combining DPPS with another dispatch policy π works as follows. In addition to empty buffers, any buffer k with $\beta_k = 0$ is also considered to be "empty". Then one uses the policy π to choose a "nonempty" buffer to work on. Once a buffer is selected, the server serves the first job of that buffer. After finishing serving that job, β corresponding to that buffer is reduced by one. Then the server updates the "nonempty" list and uses π to select the next working buffer again. This process continues until there is no "nonempty"

buffer. When this procedure stops, cycle n is finished and station i enters cycle $n+1$.

3.2 Largest Weighted Upstream Unbalanced

In this section, we will focus on slightly more restricted models. We assume that the routes of products are deterministic. Here we will use slightly different notation. Let (p,k) denote buffer k for product p. Let $Z_{p,k}(t)$ be the queue length in buffer k of product p. Let $Z^+_{p,k}(t)$ be the number of jobs of product p in the steps preceding step k (including k), thus$Z^+_{p,k}(t) = \sum_{\ell=1}^{k} Z_{p,\ell}(t)$. Let parameter β_p be weight given to product p and let $\xi_{p,k}$ be a constant which represents the desired average queue length of buffer k of product p. A Largest Weighted Up Stream Unbalanced (LWUU) policy works as follows. When station i at time t needs to decide which job to serve next, it always picks the first job of the buffer (q,ℓ) such that

$$(q,\ell) = \arg\max_{(p,k)} \beta_p \sum_{j=1}^{k} (Z_{p,j}(t) - \xi_{p,j}).$$

Note that β_p is positive number and can be chosen to reflect the importance of product p. The more important the product p, the larger β_p should be. For the $\xi_{p,k}$, one can choose any real number or simply let $\xi_{p,k}$ be 0. In this case the policy will select the class with the largest weighted number of upstream jobs. One possibility is to let $\xi_{p,k}$ equal the average queue length of class (p,k) which would have to be estimated. There are several methods to estimate it. For example, start by letting $\xi_{p,k}$ equal 0. The average queue length with this value provides an initial estimate of $\xi_{p,k}$. Simulation with this value provides a yet better estimate of $\xi_{p,k}$. After several iterations, one can obtain pretty good estimate of $\xi_{p,k}$.

3.3 Largest Weighted Total Unbalanced Policies

As in the last section, we again only consider restricted models, where the routes are assumed to be deterministic. Similar to a LWUU policy, a largest weighted total unbalance (LWTU) policy also uses imbalance information on queue lengths. However, a LWTU policy considers the imbalance not only upstream, but also downstream. Operating under a LWTU policy, a server always picks a job to process from class (q, ℓ) such that

$$(q, \ell) = \arg\max_{(p,k)} \beta_p \left[\sum_{j \le k} (Z_{p,j}(t) - \xi_{p,j}) - \sum_{j > k} (Z_{p,j}(t) - \xi_{p,j}) \right].$$

The parameters β_p, $\xi_{p,k}$ can be chosen using methods similar to those discussed in the previous section.

Chapter 4

Dynamics of Multiclass Queueing Networks and Fluid Models

In this chapter, we introduce the queueing network equations of a queueing network and the fluid model equations of the corresponding fluid model. In addition, we discuss the fluid limits connecting the queueing networks and the fluid models.

4.1 Queueing Networks Equations

The dynamics of the queueing network can be described by a process $\mathbb{X} = (A, D, T, U, Y, Z)$. The components $A = \{A(t), t \geq 0\}$, $D = \{D(t), t \geq 0\}$, $T = \{T(t), t \geq 0\}$, and $Z = \{Z(t), t \geq 0\}$ are K dimensional. For each class k, $A_k(t)$, the arrival process, denotes the number of jobs that have arrived to class k (from external and internal sources) in $[0, t]$, $D_k(t)$, the departure process, denotes the number of jobs that have departed from class k in $[0, t]$, $T_k(t)$, the server allocation process, denotes the amount of time that server $j = \sigma(k)$ has spent serving class k jobs during the interval $[0, t]$, and $Z_k(t)$, the job count process, denotes the number of jobs in class k that are waiting or being served at station j at time t. The components $Y = \{Y(t), t \geq 0\}$ and $U = \{U(t), t \geq 0\}$ are J dimensional, where J is the number of stations. For each station j, $Y_j(t)$ denotes the total amount of time that server j has been idle in the

time interval $[0,t]$ and $U_j(t)$ denotes the total number of jobs at station j that are in the buffer or being served at time t. The process Y is called the cumulative idle time process. One can check that $\mathbb{X} = (A, D, T, Y, U, Z)$ satisfies the following equations:

$$A(t) = E(t) + \sum_k \Phi^k(D(t)), \tag{4.1.1}$$

$$Z(t) = Z(0) + A(t) - D(t), \tag{4.1.2}$$

$$CT(t) + Y(t) = et, \tag{4.1.3}$$

$$Y_i(t) \text{ can increase only when } U_i(t) = 0, k = 1, ..., K, \tag{4.1.4}$$

$$\tag{4.1.5}$$

where C is the constituency matrix defined as

$$C_{ik} = \begin{cases} 1 & \text{if } k \in \mathcal{C}(i), \\ 0 & \text{otherwise,} \end{cases} \tag{4.1.6}$$

e denotes the J vector of all $1'$s. We note that T and Y are continuous, and that A, D, U and Z are right continuous with left limits. All of the variables are nonnegative in each component, and A, D, T and Y being nondecreasing. For a particular dispatch policy, we also have following equation:

$$\text{additional equations associated with the particular dispatch policy.} \tag{4.1.7}$$

By assumption, one has

$$A(0) = D(0) = T(0) = Y(0) = 0. \tag{4.1.8}$$

For a HL dispatch policy, we also have

$$V(D(t)) \leq T(t) \leq V(D(t) + e),$$

where the inequalities are componentwise and e denotes the K vector of all $1'$s.

4.2 Rate Stability

There are several definitions of stability for multiclass queueing networks. Here we are going to introduce the simplest definition of stability, rate stability. A network is said to be 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 [16] and the references there). This notion of stability was first introduced for multiclass queueing network settings in Chen [6].

Let α_k be the external arrival rate and P_{kl} be the probability that a class k job joins class l when it leaves class k. The vector $\lambda = (\lambda_1, \ldots, \lambda_K)'$ of nominal total arrival rates satisfies the following system of equations

$$\lambda_l = \alpha_l + \sum_{k=1}^{K} \lambda_k P_{kl}, \text{ for } \ell = 1, 2, ..., K. \tag{4.2.1}$$

In vector form, $\lambda = \alpha + P'\lambda$. Since P is transient, the unique solution to (4.2.1) of λ is given by $\lambda = Q\alpha$. Recall that $Q = (I - P')^{-1}$. We define the traffic intensity ρ_j for server j as

$$\rho_j = \sum_{k \in \mathcal{C}(j)} \lambda_k m_k, \quad j = 1, \ldots, J, \tag{4.2.2}$$

with ρ being the corresponding vector. Note that ρ_j is the nominal utilization of server j. When

$$\rho_j \leq 1, \quad j = 1, ..., J, \tag{4.2.3}$$

we say that the usual traffic condition is satisfied.

Recall that $D_k(t)$ denotes the number of jobs that have departed from class k in $[0, t]$.

Definition 4.2.1. The queuing network is rate stable if for each fixed initial state, with probability one,

$$\lim_{t\to\infty} \frac{D_k(t)}{t} = \lambda_k \text{ for } k = 1, ..., K. \tag{4.2.4}$$

Therefore, the queuing network is said to be *rate stable* if the throughput rate or departure rate from a class is equal to the nominal total arrival rate to that class.

4.3 Fluid Models and Fluid Limits

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

$$\bar{A}(t) = \alpha' t + P' \bar{D}(t), \quad t \geq 0, \tag{4.3.1}$$

$$\bar{Z}(t) = \bar{Z}(0) + \bar{A}(t) - \bar{D}(t), \quad t \geq 0, \tag{4.3.2}$$

$$\bar{D}_k(t) = \mu_k \bar{T}_k(t), \quad t \geq 0, \tag{4.3.3}$$

$$C\bar{T}(t) + \bar{Y}(t) = et, \quad t \geq 0, \tag{4.3.4}$$

$$\bar{Y}_j(t) \text{ increases only when } \bar{U}_j(t) = 0, \ j = 1, ..., J, \tag{4.3.5}$$

$$\text{additional equations associated with the particular dispatch policy.} \tag{4.3.6}$$

Definition 4.3.1. A fluid model is said to be *weakly stable* if for each fluid model solution $\bar{\mathbb{X}}$ with $\bar{Z}(0) = 0$, $\bar{Z}(t) = 0$ for $t \geq 0$.

The criterion for including an equation in the fluid model is that the equation is satisfied by *fluid limits*. A fluid limit of a network is obtained through a law-of-large-number procedure on the queueing network process. Note that the queueing 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 discrete 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

$$\bar{\mathbb{X}}^r(t,\omega) = r^{-1}\mathbb{X}(rt,\omega) \quad t \geq 0. \tag{4.3.7}$$

Note that again for each $r > 0$, $\bar{\mathbb{X}}^r(\cdot,\omega)$ is an element in $\mathbb{D}^{4K+2J}[0,\infty)$. The scaling in (4.3.7) is called the fluid or law-of-large-numbers scaling.

Definition 4.3.2. A function $\bar{\mathbb{X}} \in \mathbb{D}^{4K+2J}[0,\infty)$ is said to be a *fluid limit* of the queueing network if there exists a sequence $r_n \to \infty$ and a sample ω satisfying (2.2.1) such that

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

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

The existence of fluid limits is well known, see for example, Dai [10].

The introduction of fluid limits connects discrete queueing networks and fluid models. Particularly, weak stability of a fluid model implies rate stability of the corresponding discrete queueing network. Actually we have the following theorem first explicitly stated in Chen [6].

Theorem 4.3.1. *If the fluid model is weakly stable, then the corresponding discrete queueing network is rate stable.*

A similar relationship between fluid models and discrete queueing networks also holds under other definitions of stability, such as positive Harris recurrence. For addtional details, see Dai [10].

Chapter 5

Stability of Dispatch Policies

In this chapter, we explore the stability properties of DPPS, LWUU, and LWUT policies. In Section 5.1, we establish the stability of DPPS policies by showing that the fluid limits of queueing networks operating under a DPPS policy satisfy the HLPPS fluid model. In Section 5.2 and Section 5.3, we give corresponding fluid models of LWUU and LWUT policies; and then justify that the fluid limits of queueing network under LWUU and LWUT policies are the solutions of corresponding fluid models; finally, using Lyapunov functions, we show that the fluid models of LWUU and LWUT policies are stable.

5.1 Stability of DPPS Policies

Theorem 5.1.1. *A multi-class queueing network, in which the traffic intensity of each station is less than one is rate stable under any DPPS policy.*

Recall that under the HLPPS fluid model, the equation (4.3.6) is replaced by

$$\dot{T}_k(t) = \frac{Z_k(t)}{U_i(t)} \tag{5.1.1}$$

for each i such that $U_i(t) > 0$. Bramson [3] shows that HLPPS fluid models are stable under the conventional traffic intensity condition. Hence to prove Theorem 5.1.1,

using Theorem 4.3.1 we only need to show that each fluid limit of a queueing network operating under a DPPS policy satisfies the equations of the HLPPS fluid model. Actually we have following proposition.

Proposition 5.1.1. *Each fluid limit of a queueing network operating under a DPPS policy is a fluid model solution to the HLPPS fluid model.*

We finish this section by providing a proof of Proposition 5.1.1.

Proof of Proposition 5.1.1. Let $\bar{\mathbb{X}}$ be a fluid limit of the queueing network operating under a DPPS policy. To show that $\bar{\mathbb{X}}$ is a solution of the HLPPS fluid model, we need to show that the fluid limit satisfies each equation of the HLPPS fluid model. Since equation (4.3.1) to (4.3.5) are common to the fluid models of every dispatch policy, we only show that $\bar{\mathbb{X}}$ satisfies (5.1.1) of the HLPPS fluid model. Suppose that at time t, $\bar{U}_i(t) > 0$. Then there exists a δ such that for all $s \in (t, t+\delta)$, $\bar{U}_i(s) > 0$. It is enough to show that

$$\bar{T}_k(t+\delta) - \bar{T}_k(t) = \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)}\, ds. \tag{5.1.2}$$

Since $\bar{\mathbb{X}}$ is a fluid limit, there exists a sequence $\{r_n\}$ with $r_n \to \infty$ as $n \to \infty$ such that $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ u.o.c. as $n \to \infty$. Now consider the discrete queueing network. Suppose that during time period $(r_n t, r_n t + r_n \delta)$, there are $N^n - 1$ complete cycles with N^n decision points:

$$r_n t_1^n < r_n t_2^n < \ldots < r_n t_{N^n}^n. \tag{5.1.3}$$

Note that $r_n t < r_n t_1^n$ and $r_n t_{N^n}^n < r_n(t+\delta)$. Furthermore, intervals $(r_n t, r_n t_1^n)$ and $(r_n t_{N^n}^n, r_n(t+\delta))$ form two incomplete cycles. We first show that

$$\bar{T}_k(t+\delta) - \bar{T}_k(t) = \lim_{n\to\infty} \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} \frac{L_j^n}{r_n}. \tag{5.1.4}$$

Recall that $p_k(j)$ is the quota that is lost during cycle j. By Lemma 5.1.1 below,

$$D_k(r_n t^n_{N^n}) - D_k(r_n t^n_1) = \sum_{j=1}^{N^n-1} \frac{Z_k(t^n_j)}{U_i(t^n_j)} L^n_j \mu_k - \sum_{j=1}^{N^n-1} p_k(j) + r_k(0) - r_k(N^n - 1),$$

where $r_k(\cdot)$ is defined in equation (3.1.5). Dividing both sides by r_n and letting $n \to \infty$, we have

$$\lim_{n\to\infty} \frac{D_k(r_n t^n_{N^n}) - D_k(r_n t^n_1)}{r_n} = \lim_{n\to\infty} \left[\sum_{j=1}^{N^n-1} \frac{Z_k(r_n t^n_j)}{U_i(r_n t^n_j)} \frac{L^n_j}{r_n} \mu_k - \sum_{j=1}^{N^n-1} \frac{p_k(j)}{r_n} + \frac{r_k(0) - r_k(N^n - 1)}{r_n} \right].$$

We show in Lemma 5.1.2

$$\lim_{n\to\infty} \frac{1}{r_n} \sum_{j=1}^{N^n-1} p_k(j) = 0.$$

Since $r_k(0) < 1$ and $r_k(N^n - 1) < 1$, we have

$$\lim_{n\to\infty} \frac{r_k(0) - r_k(N^n - 1)}{r_n} = 0.$$

Hence, we have

$$\lim_{n\to\infty} \frac{D_k(r_n t^n_{N^n}) - D_k(r_n t^n_1)}{r_n} = \lim_{n\to\infty} \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t^n_j)}{U_i(r_n t^n_j)} \frac{L^n_j}{r_n} \mu_k. \tag{5.1.5}$$

Since $(r_n t, r_n t^n_1)$ and $(r_n t^n_{N^n}, r_n(t+\delta))$ are incomplete cycles, and the total number of class k jobs served during each period is less than $L_{max}\mu_k + 1$, we have

$$\lim_{n\to\infty} \frac{D_k(r_n t^n_1) - D_k(r_n t)}{r_n} \le \lim_{n\to\infty} \frac{L_{max}\mu_k + 1}{r_n} = 0,$$

and

$$\lim_{n\to\infty} \frac{D_k(r_n(t+\delta)) - D_k(r_n t^n_{N^n})}{r_n} \le \lim_{n\to\infty} \frac{L_{max}\mu_k + 1}{r_n} = 0.$$

So we have

$$\lim_{n\to\infty} \frac{D_k(r_n t^n_1) - D_k(r_n t)}{r_n} = 0$$

and

$$\lim_{n\to\infty} \frac{D_k(r_n(t+\delta)) - D_k(r_n t^n_{N^n})}{r_n} = 0.$$

So, we have

$$\lim_{n\to\infty} \frac{D_k(r_n t^n_{N^n}) - D_k(r_n t^n_1)}{r_n} = \lim_{n\to\infty} \frac{D_k(r_n(t+\delta)) - D_k(r_n t)}{r_n} = \bar{D}_k(t+\delta) - \bar{D}_k(t). \tag{5.1.6}$$

Equation (5.1.4) follows from (4.3.3), (5.1.5) and (5.1.6).

Next, we claim that

$$\lim_{n\to\infty} \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t^n_j)}{U_i(r_n t^n_j)} \frac{L^n_j}{r_n} = \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds. \tag{5.1.7}$$

Since $\bar{\mathbb{X}}$ is a fluid limit, for each $\epsilon > 0$, there exists an M_1 such that for all $n > M_1$,

$$\sup_{s\in[t,t+\delta]} \left| \frac{Z_k(r_n s)}{U_i(r_n s)} - \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} \right| < \frac{\epsilon}{6\delta}.$$

For $j = 0, 1, \ldots, N^n$, let

$$s^n_{j+1} = t^n_1 + \sum_{i=1}^{j} \frac{L^n_j}{r_n}. \tag{5.1.8}$$

Note that $r_n s^n_j$ is the ending time of the jth nominal cycle, whereas $r_n t^n_j$ is the ending time of the jth actual cycle in $(r_n t, r_n(t+\delta))$. It will be proven in Lemma 5.1.3 below that these times cannot differ much in fluid scaling, namely,

$$\lim_{n\to\infty} \max_{j\le N^n} \mid t^n_j - s^n_j \mid = 0.$$

Since $\bar{U}_i(s) > 0$ for $s \in [t, t+\delta]$, $\bar{Z}_k(\cdot)/\bar{U}_i(\cdot)$ is uniformly continuous over $[t, t+\delta]$. Therefore, there exists an M_2 such that for all $n > M_2$,

$$\max_{j\le N^n} \left| \frac{\bar{Z}_k(t^n_j)}{\bar{U}_i(t^n_j)} - \frac{\bar{Z}_k(s^n_j)}{\bar{U}_i(s^n_j)} \right| < \frac{\epsilon}{6\delta}.$$

Finally, by Lemma 5.1.4 below, there exists M_3 such that for all $n > M_3$,

$$\left| \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)}(s_{j+1}^n - s_j^n) - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \right| < \frac{\epsilon}{3}.$$

Thus for $n > \max\{M_1, M_2, M_3\}$, we have

$$\begin{aligned}
\left| \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} \frac{L_j^n}{r_n} - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \right| &= \left| \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} \frac{L_j^n}{r_n} - \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} \frac{L_j^n}{r_n} \right. \\
&\quad + \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} \frac{L_j^n}{r_n} - \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)} \frac{L_j^n}{r_n} + \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)} \frac{L_j^n}{r_n} - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \Bigg| \\
&\leq \left| \sum_{j=1}^{N^n-1} \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} \frac{L_j^n}{r_n} - \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} \frac{L_j^n}{r_n} \right| + \left| \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} \frac{L_j^n}{r_n} - \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)} \frac{L_j^n}{r_n} \right| \\
&\quad + \left| \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)} \frac{L_j^n}{r_n} - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \right| \\
&\leq \sup_{s \in [t, t+\delta]} \left| \frac{Z_k(r_n s)}{U_i(r_n s)} - \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} \right| \sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} + \max_{j \leq N^n} \left| \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} - \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)} \right| \sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} \\
&\quad + \left| \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)}(s_{j+1}^n - s_j^n) - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \right| \\
&\leq \frac{2\epsilon}{6\delta} \sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} + \frac{\epsilon}{3}.
\end{aligned}$$

But

$$\sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} = s_{N^n}^n - s_1^n < s_{N^n}^n - t \leq |s_{N^n}^n - t_{N^n}^n| + |t_{N^n}^n - t| \leq |s_{N^n}^n - t_{N^n}^n| + \delta,$$

and by Lemma 5.1.3,

$$\lim_{n \to \infty} \max_{j \leq N^n} |t_j^n - s_j^n| = 0,$$

we have

$$\limsup_{n \to \infty} \sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} \leq \delta.$$

Thus, for large enough n, we have

$$\sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} < 2\delta,$$

and thus,

$$\frac{\epsilon}{3\delta} \sum_{j=1}^{N^n-1} \frac{L_j^n}{r_n} + \frac{\epsilon}{3} < \epsilon.$$

Thus, (5.1.7) holds. Finally, (5.1.2) follows from (5.1.4) and (5.1.7). □

We end this section by stating the lemmas that were used in the proof of Proposition 5.1.1. We leave the proofs of these lemmas to Section 5.1.1. Recall that $p_k(\ell)$ is the class k quota lost during cycle ℓ. Let t_n and t_m be two decision points with $m > n$. Our first lemma relates the number of jobs served with the nominal number of jobs allocated and the lost quotas.

Lemma 5.1.1. *Assume that station i is nonempty throughout (t_n, t_m). Then,*

$$D_k(t_m) - D_k(t_n) = \sum_{\ell=n}^{m-1} \frac{Z_k(t_\ell)}{U_i(t_\ell)} L_\ell \, \mu_k - \sum_{\ell=n}^{m-1} p_k(\ell) + r_k(n-1) - r_k(m-1). \quad (5.1.9)$$

The next lemma shows that the cumulative lost quotas are negligible in fluid scaling.

Lemma 5.1.2.

$$\lim_{n\to\infty} \sum_{j=1}^{N^n} \frac{p_k(j)}{r_n} = 0.$$

The next lemma shows that the difference between the end times of the nominal and actual periods is negligible in fluid scaling.

Lemma 5.1.3. *Let t_j^n and s_j^n be defined as in (5.1.3) and (5.1.8).*

$$\lim_{n\to\infty} \max_{j\le N^n} |t_j^n - s_j^n| = 0.$$

Our last lemma shows that a Riemann type of sum is close to the desired integral. One main issue is that the s_j's may lie outside of $(t, t+\delta)$.

Lemma 5.1.4. *Let s_j^n be defined as in (5.1.8).*

$$\lim_{n\to\infty} \left| \sum_{j=1}^{N^n-1} \frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)}(s_{j+1}^n - s_j^n) - \int_t^{t+\delta} \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} ds \right| = 0. \tag{5.1.10}$$

5.1.1 Proofs of Lemmas 5.1.1-5.1.4

In this section, we prove Lemmas 5.1.1-5.1.4. We first prove Lemma 5.1.1.

Proof of Lemma 5.1.1. If $Z_k(t_{\ell+1}) > 0$, then $p_k(\ell) = 0$ and

$$\lfloor q_k(\ell) \rfloor = D_k(t_{\ell+1}) - D_k(t_\ell). \tag{5.1.11}$$

If $Z_k(t_{\ell+1}) = 0$, then

$$p_k(\ell) = q_k(\ell) - \left(D_k(t_{\ell+1}) - D_k(t_\ell)\right). \tag{5.1.12}$$

Hence, the number of class k jobs served in cycle ℓ, $D_k(t_{\ell+1}) - D_k(t_\ell)$, is equal to $\lfloor q_k(\ell) - p_k(\ell) \rfloor$, and thus

$$D_k(t_m) - D_k(t_n) = \sum_{\ell=n}^{m-1} \lfloor q_k(\ell) - p_k(\ell) \rfloor.$$

On the other hand, by equations (3.1.4) and (3.1.5), we have

$$r_k(\ell) = (q_k(\ell) - \lfloor q_k(\ell) \rfloor) 1_{\{Z_k(t_{\ell+1})>0\}}.$$

We claim that

$$r_k(\ell) = q_k(\ell) - p_k(\ell) - \lfloor q_k(\ell) - p_k(\ell) \rfloor. \tag{5.1.13}$$

To show this, we consider two cases.

Case 1: $Z_k(t_{\ell+1}) = 0$. Then we have $r_k(\ell) = 0$. By (5.1.12)

$$q_k(\ell) - p_k(\ell) = D_k(t_{\ell+1}) - D_k(t_\ell).$$

So $p_k(\ell) - q_k(\ell)$ is an integer, and thus $q_k(\ell) - p_k(\ell) - \lfloor q_k(\ell) - p_k(\ell) \rfloor = 0$. Therefore, (5.1.13) holds.

Case 2: $Z_k(t_{\ell+1}) > 0$. In this case, (5.1.13) follows from the fact that $p_k(\ell) = 0$ and $r_k(\ell) = q_k(\ell) - \lfloor q_k(\ell) \rfloor$. Hence,

$$D_k(t_m) - D_k(t_n) = \sum_{\ell=n}^{m-1} \left[q_k(\ell) - p_k(\ell) - r_k(\ell) \right].$$

Using equation (3.1.4), we have

$$\begin{aligned} D_k(t_m) - D_k(t_n) &= \sum_{\ell=n}^{m-1} \left[r_k(\ell-1) + \frac{Z_k(t_\ell)}{U_i(t_\ell)} L_\ell \mu_k - p_k(\ell) - r_k(\ell) \right] \\ &= \sum_{\ell=n}^{m-1} \frac{Z_k(t_\ell)}{U_i(t_\ell)} L_\ell \mu_k - \sum_{\ell=n}^{m-1} p_k(\ell) + r_k(n-1) - r_k(m-1), \end{aligned}$$

proving the lemma. □

For Lemmas 5.1.2-5.1.4, recall that r_n is so chosen that $r_n \to \infty$ and $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ u.o.c. as $n \to \infty$. It is assumed that $\bar{U}_i(s) > 0$ for $s \in [t, t+\delta]$, and that there are N^n complete cycles in time interval $(r_n t, r_n(t+\delta))$. Before we prove the next lemma, we first start with a few lemmas that will be used in the proofs of Lemmas 5.1.2-5.1.4.

Lemma 5.1.5.

$$\limsup_{n\to\infty} \frac{D_k(r_n t_1^n)}{r_n} < \infty, \tag{5.1.14}$$

$$\liminf_{n\to\infty} \frac{N^n}{r_n} > 0, \tag{5.1.15}$$

$$\limsup_{n\to\infty} \frac{N^n}{r_n} < \infty. \tag{5.1.16}$$

Proof. Note that $D_k(r_n t_1^n) \leq D_k(r_n(t+\delta))$, we have

$$\limsup_{n\to\infty} \frac{D_k(r_n t_1^n)}{r_n} \leq \lim_{n\to\infty} \frac{D_k(r_n(t+\delta))}{r_n} = \bar{D}_k(t+\delta) < \infty,$$

thus proving (5.1.14).

To prove (5.1.15), we note that $\bar{U}_i(s) > 0$ for $s \in (t, t+\delta)$. Thus, for large enough n, $U_i(s) > 0$ for $s \in (r_n t, r_n(t+\delta))$, and therefore server i is busy during the entire interval. Hence, at least one class of jobs, say class k, receives at least $r_n\delta/|\mathcal{C}(i)|$ amount of time from the server during $(r_n t, r_n(t+\delta))$, where $|\mathcal{C}(i)|$ is the number of classes at station i. Recall that $\xi_k(\ell)$ is the service time for the ℓth class k job. We have,

$$\sum_{\ell=D_k(r_n t)}^{D_k(r_n(t+\delta))+1} \xi_k(\ell) \geq \frac{r_n\delta}{|\mathcal{C}(i)|}. \tag{5.1.17}$$

Dividing both side by r_n and taking the liminf, we can get

$$\liminf \frac{1}{r_n} \sum_{\ell=D_k(r_n t)}^{D_k(r_n(t+\delta))+1} \xi_k(\ell) \geq \frac{\delta}{|\mathcal{C}(i)|}. \tag{5.1.18}$$

Since $\lim_{n\to\infty} D_k(r_n t)/r_n$ and $\lim_{n\to\infty} D_k(r_n(t+\delta))/r_n$ exist, by Lemma 5.1.8, we have

$$\lim_{n\to\infty} \frac{D_k(r_n(t+\delta)) - D_k(r_n t)}{r_n} > 0.$$

Now because the number of class k jobs served during a cycle is at most $L_{max}\mu_k$, we have

$$(N^n + 1)L_{max}\mu_k > D_k(r_n(t+\delta)) - D_k(r_n t).$$

Thus,

$$\liminf_{n\to\infty} \frac{N^n}{r_n} > 0.$$

To show (5.1.16), we note that the number of cycles is fewer than the total number of jobs served during $(r_n t, r_n t + r_n \delta)$. So we have

$$\limsup_{n\to\infty} \frac{N^n}{r_n} \le \lim_{n\to\infty} \frac{1}{r_n} \sum_{k\in\mathcal{C}(i)} [D_k(r_n(t+\delta)) - D_k(r_n t)] = \sum_{k\in\mathcal{C}(i)} [\bar{D}_k(t+\delta) - \bar{D}_k(t)] < \infty.$$

□

The next lemma is used in the proof of Lemma 5.1.2 only.

Lemma 5.1.6. *For any $\epsilon > 0$, there exists an $N > 0$ such that for all $n > N$ and $s \in (t, t+\delta)$,*

$$1_{\{Z_k(r_n s) < L_{max}\mu_k\}} \le 1_{\{\bar{Z}_k(s) < \epsilon\}},$$

where $1_{\{.\}}$ is an indicator function.

Proof. Since

$$\lim_{r_n\to\infty} \frac{Z_k(r_n s)}{r_n} = \bar{Z}_k(s), \quad \text{u.o.c.},$$

there exists an M such that for all $n > M$,

$$\sup_{s\in(t,t+\delta)} \left| \frac{Z_k(r_n s)}{r_n} - \bar{Z}_k(s) \right| < \frac{\epsilon}{2}.$$

Thus, for any $s \in (t, t+\delta)$ with $\bar{Z}_k(s) \ge \epsilon$,

$$Z_k(r_n s) \ge r_n \epsilon/2, \quad \text{for all} \quad n > M.$$

Choose $N > M$ such that for all $n > N$, $r_n\epsilon/2 > L_{max}\,\mu_k$. Thus for all $n > N$, $Z_k(r_n s) \ge L_{max}\mu_k$. So

$$1_{\{Z_k(r_n s) \ge L_{max}\,\mu_k\}} \ge 1_{\{\bar{Z}_k(s) \ge \epsilon\}}.$$

Therefore for all $n > N$ and $s \in (t, t+\delta)$, we have

$$1_{\{Z_k(r_n s) < L_{max}\mu_k\}} \le 1_{\{\bar{Z}_k(s) < \epsilon\}}.$$

□

Proof of Lemma 5.1.2. Because $L_j < L_{max}$, during cycle j,

$$q_k(j) \leq 1 + \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} L_j \mu_k \leq 1 + L_{max}\mu_k.$$

Let $C = 1 + L_{max}\mu_k$. Then when $Z_k(r_n t_j^n) \geq C$, the quota lost during cycle j is zero. On the other hand, if $Z_k(r_n t_j^n) < C$, the quota lost during cycle j is at most

$$\frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} L_{max}\mu_k.$$

Hence

$$p_k(j) \leq \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} L_{max}\mu_k 1_{\{Z_k(r_n t_j^n) < C\}}.$$

Therefore we have

$$\sum_{j=1}^{N^n} \frac{p_k(j)}{r_n} \leq \frac{1}{r_n} \sum_{j=1}^{N^n} \frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} L_{max}\mu_k 1_{\{Z_k(r_n t_j^n) < C\}}.$$

By Lemma 5.1.6, for any $\epsilon > 0$, there exists an $N_1 > 0$ such that for $n > N_1$ and $s \in (t, t+\delta)$,

$$1_{\{Z_k(r_n s) < C\}} \leq 1_{\{\bar{Z}_k(s) < \epsilon\}}.$$

Since $\bar{\mathbb{X}}$ is a fluid limit, there exists an N_2 such that for all $n > N_2$,

$$\sup_{s \in (t, t+\delta)} \left| \frac{Z_k(r_n s)}{U_i(r_n s)} - \frac{\bar{Z}_k(s)}{\bar{U}_i(s)} \right| < \epsilon. \tag{5.1.19}$$

Finally, by (5.1.16), there exists a $B > 0$ and an $N_3 > 0$ such that for all $n > N_3$,

$\frac{N^n}{r_n} < B$. Now, for $n > \max\{N_1, N_2, N_3\}$, we have

$$\begin{aligned}
&\frac{1}{r_n}\sum_{j=1}^{N^n}\frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)}L_{max}\mu_k 1_{\{Z_k(r_n t_j^n)<C\}} \\
&= \frac{L_{max}\mu_k}{r_n}\sum_{j=1}^{N^n}\left[\frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} - \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)} + \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)}\right]1_{\{Z_k(r_n t_j^n)<C\}} \\
&\leq \frac{L_{max}\mu_k}{r_n}\sum_{j=1}^{N^n}\left|\frac{Z_k(r_n t_j^n)}{U_i(r_n t_j^n)} - \frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)}\right| + \frac{L_{max}\mu_k}{r_n}\sum_{j=1}^{N^n}\frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)}1_{\{Z_k(r_n t_j^n)<C\}} \\
&\leq \frac{N^n}{r_n}L_{max}\mu_k\epsilon + \frac{L_{max}\mu_k}{r_n}\sum_{j=1}^{N^n}\frac{\bar{Z}_k(t_j^n)}{\bar{U}_i(t_j^n)}1_{\{\bar{Z}_k(t_j^n)<\epsilon\}} \\
&\leq \frac{N^n}{r_n}L_{max}\mu_k\epsilon + \frac{L_{max}\mu_k\epsilon}{\min_{s\in(t,t+\delta)}\bar{U}_i(s)}\frac{N^n}{r_n} \\
&\leq BL_{max}\mu_k\left(1 + \frac{1}{\min_{s\in[t,t+\delta]}\bar{U}_i(s)}\right)\epsilon.
\end{aligned}$$

□

Now we turn to the proof of Lemma 5.1.3. First, we introduce some additional notation and an additional lemma.

Let $F_k^n(j) = D_k(r_n t_{j+1}^n) - D_k(r_n t_1^n)$ be the number of class k jobs served from time $r_n t_1^n$ to $r_n t_{j+1}^n$. By Lemma 5.1.1, we have

$$F_k^n(l) = \sum_{j=1}^{l}\frac{Z_k(t_j)}{U_i(r_n t_j^n)}L_j\mu_k + r_k(r_n t_0^n) - r_k(r_n t_l^n) - \sum_{j=1}^{l}p_k(j). \tag{5.1.20}$$

Multiplying both sides of (5.1.20) by m_k and summing up all classes in $\mathcal{C}(i)$, we have

$$\sum_{k\in\mathcal{C}(i)}F_k^n(l)m_k = \sum_{j=1}^{l}L_j + \sum_{k\in\mathcal{C}(i)}[r_k(r_n t_0^n) - r_k(r_n t_l^n)]m_k - \sum_{j=1}^{l}m_k p_k(j). \tag{5.1.21}$$

Lemma 5.1.7.

$$\lim_{n\to\infty}\max_{j\leq N^n}\frac{1}{r_n}\left|\sum_{\ell=D_k(r_n t_1^n)}^{D_k(r_n t_j^n)}\xi_k(l) - F_k^n(j-1)m_k\right| = 0.$$

Proof. Let $\hat{\xi}_k(l) = \xi_k(l) - m_k$. Then

$$\max_{j\le N^n}\left|\sum_{\ell=D_k(r_nt_1^n)}^{D_k(r_nt_j^n)}\xi_k(l) - F_k^n(j-1)m_k\right| = \max_{j\le N^n}\left|\sum_{\ell=D_k(r_nt_1^n)}^{D_k(r_nt_1^n)+F_k^n(j-1)}\hat{\xi}_k(l)\right|.$$

By Lemma 5.1.5, we

$$\begin{aligned}
&\limsup_{n\to\infty}\frac{D_k(r_nt_1^n)}{r_n} < +\infty,\\
&\limsup_{n\to\infty}\frac{F_k^n(N^n-1)}{r_n} \le \lim_{n\to\infty}\frac{D_k(r_n(t+\delta))}{r_n} < +\infty,\\
&\lim_{n\to\infty} N^n = \infty.
\end{aligned}$$

So the conditions of Lemma 5.1.9 are satisfied, and thus

$$\lim_{n\to\infty}\max_{j\le N^n}\left|\sum_{\ell=D_k(r_nt_1^n)}^{D_k(r_nt_1^n)+F_k^n(j-1)}\frac{\hat{\xi}_k(l)}{r_n}\right| = 0,$$

and the lemma is proved. □

Proof of Lemma 5.1.3. Recall that for $j = 1, \dots, N^n$,

$$r_ns_j^n = r_nt_1^n + \sum_{i=1}^{j-1} L_i$$

is the end point of the jth nominal cycle, and $r_nt_j^n$ is the end point of the jth actual cycle in $(r_nt, r_n(t+\delta))$. Thus,

$$r_n(t_j^n - t_1^n) = \sum_{k\in\mathcal{C}(i)}\sum_{l=D_k(r_nt_1^n)+1}^{D_k(r_nt_1^n)+F_k^n(j-1)}\xi_k(l),$$

and

$$r_n(t_j^n - s_j^n) = \sum_{k\in\mathcal{C}(i)}\sum_{l=D_k(r_nt_1^n)+1}^{D_k(r_nt_1^n)+F_k^n(j-1)}\xi_k(l) - \sum_{i=1}^{j-1}L_i.$$

By (5.1.21), we have

$$\sum_{\ell=1}^{j-1}L_\ell = \sum_{k\in\mathcal{C}(i)}F_k^n(j-1)m_k - \sum_{k\in\mathcal{C}(i)}[r_k(t_0) - r_k(t_{j-1})]m_k + \sum_{k\in\mathcal{C}(i)}\sum_{\ell=1}^{j-1}m_kp_k(t_\ell).$$

Therefore,

$$t_j^n - s_j^n = \sum_{k\in\mathcal{C}(i)} \left(\sum_{l=D_k(r_n t_1^n)+1}^{D_k(r_n t_1^n)+F_k^n(j-1)} \frac{\xi_k(l)}{r_n} - \frac{F_k(j-1)m_k}{r_n} \right) + \sum_{k\in\mathcal{C}(i)} \frac{[r_k(t_0) - r_k(t_{j-1})]m_k}{r_n} - \sum_{k\in\mathcal{C}(i)} \sum_{\ell=1}^{j-1} \frac{m_k p_k(t_\ell)}{r_n}.$$

It follows that

$$\max_{j\le N^n} |t_j^n - s_j^n| \le \max_{j\le N^n} \sum_{k\in\mathcal{C}(i)} \left| \sum_{l=D_k(r_n t_1^n)+1}^{D_k(r_n t_1^n)+F_k^n(j-1)} \frac{\xi_k(l)}{r_n} - \frac{F_k^n(j-1)m_k}{r_n} \right| + \frac{1}{r_n} \sum_{k\in\mathcal{C}(i)} m_k + \max_{j\le N^n} \sum_{k\in\mathcal{C}(i)} \sum_{\ell=1}^{j-1} \frac{m_k p_k(t_\ell)}{r_n}.$$

The first term on the right converges to zero by Lemma 5.1.7. Clearly, the second term converges to zero as well. Since

$$\max_{j\le N^n} \sum_{k\in\mathcal{C}(i)} \sum_{\ell=1}^{j-1} \frac{m_k p_k(t_\ell)}{r_n} \le \sum_{k\in\mathcal{C}(i)} m_k \sum_{\ell=1}^{N^n} \frac{p_k(t_\ell)}{r_n},$$

and by Lemma 5.1.2,

$$\lim_{n\to\infty} \sum_{j=1}^{N^n} \frac{p_k(j)}{r_n} = 0$$

for each class k, we have

$$\lim_{n\to\infty} \max_{j\le N^n} \sum_{k\in\mathcal{C}(i)} \sum_{\ell=1}^{j-1} \frac{m_k p_k(t_\ell)}{r_n} = 0.$$

Thus, the third term also converges to zero. Hence we have $\lim_{n\to\infty} \max_{j\le N^n} |t_j^n - s_j^n| = 0$, proving the lemma. □

Proof of Lemma 5.1.4. Let $m = \max\{j \le N^n : s_j^n < t+\delta\}$. Then, $t, s_1^n, s_2^n, \ldots, s_m^n, t+\delta$ forms a partition of $[t, t+\delta]$. We need to show that the norm of this partition

converge to 0, namely,

$$\lim_{n\to\infty} \max_{1<j\le m} (s_j^n - s_{j-1}^n) = 0 \quad \text{a.s.}, \tag{5.1.22}$$

$$\lim_{n\to+\infty} (s_1^n - t) = 0 \quad \text{a.s.}, \tag{5.1.23}$$

$$\lim_{n\to+\infty} (t + \delta - s_m^n) = 0 \quad \text{a.s.} \tag{5.1.24}$$

Since

$$\max_{1<j\le m} (s_j^n - s_{j-1}^n) = \max_{1<j\le m} \frac{L_j^n}{r_n} \le \frac{L_{max}}{r_n} \to 0 \text{ as } n \to \infty,$$

(5.1.22) is satisfied for every sample path. Now

$$\lim_{n\to\infty} (t_1^n - t) \le \lim_{n\to\infty} \sum_{k\in\mathcal{C}(i)} \frac{1}{r_n} \sum_{\ell=D_k(r_n t)}^{D_k(r_n t_1^n)} \xi_k(\ell).$$

Since $(r_n t, r_n t_1^n)$ is a partial cycle, the total number of class k jobs served during this period is less than $L_{max}\mu_k + 1$. So

$$\frac{D_k(r_n t_1^n) - D_k(r_n t)}{r_n} < \frac{L_{max}\mu_k + 1}{r_n} \to 0, \quad \text{as} \quad n \to \infty.$$

Hence by Lemma 5.1.8 below,

$$\lim_{n\to\infty} \frac{1}{r_n} \sum_{\ell=D_k(r_n t)}^{D_k(r_n t_1^n)} \xi_k(\ell) = 0.$$

So (5.1.23) is true. Finally, if $m < N^n$, then

$$r_n(t+\delta) - r_n s_m^n < r_n s_{m+1}^n - r_n s_m^n < L_{max};$$

if $m = N^n$, then

$$r_n(t+\delta) - r_n s_m^n < r_n t_{N^n}^n - r_n s_{N^n}^n.$$

But $\lim_{n\to\infty} L_{max}/r_n = 0$ and by Lemma 5.1.3, $\lim_{n\to+\infty}(t^n_{N^n} - s^n_{N^n}) = 0$. So in either case we have (5.1.24). Thus,

$$\lim_{n\to+\infty}\left|\frac{\bar{Z}_k(t)}{\bar{U}_i(t)}(s^n_1 - t) + \sum_{j=1}^{m-1}\frac{\bar{Z}_k(s^n_j)}{\bar{U}_i(s^n_j)}(s^n_{j+1} - s^n_j) + \frac{\bar{Z}_k(s^n_m)}{\bar{U}_i(s^n_m)}(t+\delta - s^n_m) - \int_t^{t+\delta}\frac{\bar{Z}_k(s)}{\bar{U}_i(s)}ds\right| = 0. \tag{5.1.25}$$

Because $\bar{Z}_k(t)/\bar{U}_i(t)$ is bounded by 1, from (5.1.23) and (5.1.24), we have

$$\lim_{n\to+\infty}\frac{\bar{Z}_k(t)}{\bar{U}_i(t)}(s^n_1 - t) = 0 \quad a.s.$$

$$\lim_{n\to+\infty}\frac{\bar{Z}_k(s^n_m)}{\bar{U}_i(s^n_m)}(t+\delta - s^n_m) = 0 \quad a.s.$$

So we can delete the first and last terms in the left side of equation (5.1.25) and obtain

$$\lim_{n\to+\infty}\left|\sum_{j=1}^{m-1}\frac{\bar{Z}_k(s^n_j)}{\bar{U}_i(s^n_j)}(s^n_{j+1} - s^n_j) - \int_t^{t+\delta}\frac{\bar{Z}_k(s)}{\bar{U}_i(s)}ds\right| = 0. \tag{5.1.26}$$

On the other hand,

$$\sum_{j=m}^{N^n}\frac{\bar{Z}_k(s^n_j)}{\bar{U}_i(s^n_j)}(s^n_{j+1} - s^n_j) < s^n_{N^n} - s^n_m.$$

If $m = N^n$, then $s^n_{N^n} - s^n_m = 0$, otherwise

$$\begin{aligned} s^n_{N^n} - s^n_m &= s^n_{N^n} - (t+\delta) + (t+\delta) - s^n_m \\ &< |s^n_{N^n} - t^n_{N^n}| + |s^n_{m+1} - s^n_m| \\ &< |s^n_{N^n} - t^n_{N^n}| + \frac{L_{max}}{r_n}. \end{aligned}$$

But $\lim_{n\to\infty}|s^n_{N^n} - t^n_{N^n}| = 0$ and $\lim_{n\to+\infty} L_{max}/r_n = 0$, so

$$\lim_{n\to+\infty}(s^n_{N^n} - s^n_m) = 0.$$

Thus

$$\lim_{n\to+\infty}\sum_{j=m}^{N^n}\frac{\bar{Z}_k(s^n_j)}{\bar{U}_i(s^n_j)}(s^n_{j+1} - s^n_j) = 0. \tag{5.1.27}$$

So by (5.1.26) and (5.1.27), we have

$$\lim_{n\to+\infty}\left|\sum_{j=1}^{N^n-1}\frac{\bar{Z}_k(s_j^n)}{\bar{U}_i(s_j^n)}(s_{j+1}^n-s_j^n)-\int_t^{t+\delta}\frac{\bar{Z}_k(s)}{\bar{U}_i(s)}ds\right|=0. \tag{5.1.28}$$

□

5.1.2 Some Additional Lemmas

In this section, we present a few lemmas that concern a general sequence of real numbers that possesses a long-run average. Throughput this section, assume that $\{\xi(i):i=1,\dots,\}$ is a sequence of real numbers satisfying

$$\frac{1}{n}(\xi(1)+\ldots+\xi(n))\to\bar{\xi}\quad\text{ as } n\to\infty \tag{5.1.29}$$

for some real number $\bar{\xi}$.

Our first lemma was used in the proof of Lemma 5.1.4, again in the proof of Lemma 5.1.9. The elementary proof can be found in, for example, Jennings [21].

Lemma 5.1.8. *Let f_1,f_2 and g be three maps from $\mathbb{Z}_+$ to $\mathbb{Z}_+$ satisfying*

$$\limsup_{n\to+\infty}\frac{f_1(n)}{g(n)}<+\infty,\quad \lim_{n\to+\infty}\frac{f_2(n)}{g(n)}=0,\quad \textit{and}\ \lim_{n\to+\infty}g(n)=+\infty.$$

Then

$$\lim_{n\to\infty}\frac{1}{g(n)}\sum_{=f_1(n)+1}^{f_1(n)+f_2(n)}\xi(j)=0. \tag{5.1.30}$$

In the proof of Lemma 5.1.7, we made a critical use of the following lemma. The proof of this lemma is the main content of this section.

Lemma 5.1.9. *Assume that $\bar{\xi}=0$ in (5.1.29). Let f and h be two maps from $\mathbb{Z}_+$ to $\mathbb{Z}_+$, and g be a map from $\mathbb{Z}_+^2$ to $\mathbb{Z}_+$. Let $\{r_n\}$ be a sequence with $r_n\to\infty$ as $n\to\infty$.*

Suppose that

$$\begin{aligned}
&\lim_{n\to\infty} h(n) = \infty, \\
&\limsup_{n\to\infty} \frac{f(n)}{r_n} < \infty, \\
&\limsup_{n\to\infty} \frac{g(h(n),n)}{r_n} < \infty,
\end{aligned}$$

and for each fixed n, $g(m,n)$ is nondecreasing function in m. Then,

$$\lim_{n\to\infty} \frac{\max_{m\le h(n)} |\sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)|}{r_n} = 0. \tag{5.1.31}$$

To prove Lemma 5.1.9, we make use the following lemma. The proof can be found, for example, in Gut[17].

Lemma 5.1.10. *Let $S(n) = \sum_{i=1}^{n} \xi(i)$. Then*

$$\begin{aligned}
\lim_{m\to+\infty} \frac{\max_{n\le m} S(n)}{m} &= \bar{\xi}^{+}, \\
\lim_{m\to+\infty} \frac{\min_{n\le m} S(n)}{m} &= \bar{\xi}^{-},
\end{aligned}$$

where for a real number x, x^+ denotes the positive part of x and x^- denotes the negative part of x.

Proof of Lemma 5.1.9. Since

$$\begin{aligned}
&\frac{\max_{m\le h(n)} |\sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)|}{r_n} \\
= \quad &\max\left\{ \frac{\max_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)}{r_n}, \frac{-\min_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)}{r_n} \right\},
\end{aligned}$$

we only need to show

$$\limsup_{n\to\infty} \frac{\max_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)}{r_n} \le 0 \quad a.s. \tag{5.1.32}$$

and

$$\liminf_{n\to\infty} \frac{\min_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)}{r_n} \ge 0 \quad a.s. \tag{5.1.33}$$

Because $g(m,n)$ is nondecreasing with respect to m, we have

$$\max_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell) \le \max_{m\le g(h(n),n)} \sum_{\ell=f(n)}^{f(n)+m} \xi(\ell).$$

Thus,

$$\begin{aligned}
& \frac{\max_{m\le h(n)} \sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)}{r_n} \\
\le\ & \frac{\max_{m\le g(h(n),n)} \sum_{\ell=f(n)}^{f(n)+m} \xi(\ell)}{r_n} \\
=\ & \frac{\max_{m\le g(h(n),n)} \sum_{\ell=1}^{f(n)+m} \xi(\ell) - \sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n} \\
=\ & \frac{\max_{m\le g(h(n),n)} \sum_{\ell=1}^{f(n)+m} \xi(\ell)}{r_n} - \frac{\sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n} \\
\le\ & \frac{\max_{m\le g(h(n),n)+f(n)} \sum_{\ell=1}^{m} \xi(\ell)}{r_n} - \frac{\sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n} \\
\le\ & \frac{\max_{m\le g(h(n),n)+f(n)+r_n} \sum_{\ell=1}^{m} \xi(\ell)}{r_n} - \frac{\sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n} \\
=\ & \frac{\max_{m\le g(h(n),n)+f(n)+r_n} \sum_{\ell=1}^{m} \xi(\ell)}{g(h(n),n)+f(n)+r_n} \left(1 + \frac{g(h(n),n)+f(n)}{r_n}\right) - \frac{\sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n}.
\end{aligned}$$

Since $\lim_{n\to\infty}(g(h(n),n)+f(n)+r_n) = +\infty$, by Lemma 5.1.10, we have

$$\lim_{n\to\infty} \frac{\max_{m\le g(h(n),n)+f(n)+r_n} \sum_{\ell=1}^{m} \xi(\ell)}{g(h(n),n)+f(n)+r_n} = 0.$$

Together with the condition

$$\limsup_{n\to} \frac{g(h(n),n)+f(n)}{r_n} < +\infty,$$

we have

$$\lim_{n\to\infty} \frac{\max_{m\le g(h(n),n)+f(n)+r_n} \sum_{\ell=1}^{m} \xi(\ell)}{g(h(n),n)+f(n)+r_n} \left(1 + \frac{g(h(n),n)+f(n)}{r_n}\right) = 0.$$

From Lemma 5.1.8, we have

$$\lim_{n\to\infty} \frac{\sum_{\ell=1}^{f(n)} \xi(\ell)}{r_n} = 0.$$

Thus (5.1.32) is true. Similarly, we can show (5.1.33) is also true. So

$$\lim_{n\to\infty} \frac{\max_{m\le h(n)} |\sum_{\ell=f(n)}^{f(n)+g(m,n)} \xi(\ell)|}{g(h(n),n)} = 0 \quad a.s. \tag{5.1.34}$$

□

5.2 Stability of LWUU Policies

In this section, we will show that networks with deterministic routing under LWUU policies are stable. As previous sections, we will prove this by showing that fluid models are stable. The corresponding fluid model is following.

$$\bar{Z}_{p,k}(t) = \bar{Z}_{p,k}(0) + \bar{A}_{p,k}(t) - \bar{D}_{p,k}(t), p = 1,2,...P, k = 1,2,...K_p, \tag{5.2.1}$$

$$\bar{A}_{p,k}(t) = \bar{D}_{p,k-1}(t), p = 1,2,...P, k = 1,2,...K_p, \tag{5.2.2}$$

$$C\bar{T}(t) + \bar{Y}(t) = et, \quad t \ge 0, \tag{5.2.3}$$

$$\bar{Y}_j(t) \text{ increases only when } \bar{U}_j(t) = 0,\ j = 1,...,J, \tag{5.2.4}$$

$$\dot{\bar{T}}_{p,k}(t) = 0 \text{ if } \exists (q,l) \in \mathcal{C}(j) \text{ s.t. } \beta_p \bar{Z}^+_{p,k}(t) < \beta_q \bar{Z}^+_{q,l}(t) \text{ and } \bar{Z}_{q,l}(t) > 0, \tag{5.2.5}$$

where $\bar{Z}^+_{q,l}(t) = \sum_{k=1}^{l} \bar{Z}_{q,k}(t)$.

Proposition 5.2.1. *The fluid limits of queueing networks under LWUU policies satisfy the above fluid equations.*

Proof. We only check equation (5.2.5). Verification of other equations is standard. Let $\bar{X}$ be a fluid limit under a LWUU policy. Let $r_n \to \infty$ be a corresponding sequence such that $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ as $n \to \infty$. For any time t, suppose that there exists class (q,l) such that $\beta_p \bar{Z}^+_{p,k}(t) < \beta_q \bar{Z}^+_{q,l}(t)$ and $\bar{Z}_{q,l}(t) > 0$. Let

$$\epsilon = \min\{\beta_q \bar{Z}^+_{q,l}(t) - \beta_p \bar{Z}^+_{p,k}(t), \quad \bar{Z}_{q,l}(t)\}.$$

Then by the continuity of $\bar{Z}(t)$, there exists δ such that for all $s \in (t-\delta, t+\delta)$, $\beta_q \bar{Z}^+_{q,l}(s) - \beta_p \bar{Z}^+_{p,k}(s) > \epsilon/2$, and $\bar{Z}_{q,l}(s) > \epsilon/2$. Since $\bar{Z}^{r_n}(\cdot) \to \bar{Z}(\cdot)$ u.o.c. and $r_n \to \infty$ as $n \to \infty$, there exists an N such that for all $n > N$,

$$\sup_{0 \le s \le t+\delta} |\beta_p Z^+_{p,k}(r_n s)/r_n - \beta_p \bar{Z}^+_{p,k}(s)| < \epsilon/8,$$

$$\sup_{0 \le s \le t+\delta} |\beta_q Z^+_{q,l}(r_n s)/r_n - \beta_q \bar{Z}^+_{q,l}(s)| < \epsilon/8,$$

and

$$\sup_{0 \le s \le t+\delta} |Z^+_{q,l}(r_n s)/r_n - \bar{Z}_{q,l}(s)| < \epsilon/4.$$

Hence for $s \in (t-\delta, t+\delta)$,

$$\beta_q Z^+_{q,l}(r_n s)/r_n - \beta_p Z^+_{p,k}(r_n s)/r_n > \epsilon/4,$$

and

$$Z^+_{q,l}(r_n s)/r_n > \epsilon/4.$$

And thus for $s \in (r_n(t-\delta), r_n(t+\delta))$,

$$\beta_q (Z^+_{q,l}(s) - \xi^+_{q,l}) - \beta_p (Z^+_{p,k}(s) - \xi^+_{p,k}) > r_n \epsilon/4 - (\beta_q \xi^+_{q,l} - \beta_p \xi^+_{p,k}),$$

and

$$Z^+_{q,l}(s) > r_n \epsilon/4.$$

So for large enough n, we have

$$\beta_q(Z^+_{q,l}(s) - \xi^+_{q,l}) - \beta_p(Z^+_{p,k}(s) - \xi^+_{p,k}) > 0.$$

Thus by the definition of LWUU policies, the server will never work for class (p,k) during $(r_n(t-\delta), r_n(t+\delta))$. So

$$T_{p,k}(r_n(t+\delta)) - T_{p,k}(r_n(t-\delta)) = 0.$$

Divided by r_n and let $n \to \infty$ we have

$$\bar{T}_{p,k}(t+\delta) - \bar{T}_{p,k}(t-\delta) = 0.$$

Hence

$$\dot{\bar{T}}_{p,k}(t) = 0.$$

□

Let $f_{p,k}(t) = \beta_p \bar{Z}^+_{p,k}(t)$ and $f(t) = \max_{p=1,2,..P,k=1,2,...k_p} f_{p,k}(t)$. Since each $f_{p,k}$ is a linear function of $\bar{Z}(t)$, f is a piecewise linear Lyapunov function of $\bar{Z}(t)$. One can check that $f(t)$ is a Lipschitz function of t, and is thus absolutely continuous. In this section, t is said to be regular if both $\mathbb{X}$ and f are differentiable at t. Whenever a derivative is used at time t, t is assumed to be a regular point.

Lemma 5.2.1. *For a regular point t, if*

$$f(t) = \max_{p=1,2,..P,k=1,2,...k_p} f_{p,k}(t),$$

$\dot{f}_{p,k}(t) = \dot{f}(t)$.

Proof. Let $g(t) = f(t) - f_{p,k}(t)$. Then $g(t) \geq 0$. If for a regular point t, $f(t) = \max_{p=1,2,..P,k=1,2,...k_p} f_{p,k}(t)$, then $g(t) = 0$. Hence t is minimum point. Since t is regular point for both $f(\cdot)$ and $f_{p,k}(\cdot)$ and hence is also regular point of $g(\cdot)$, we have $\dot{g}(t) = 0$. Thus $\dot{f}_{p,k}(t) = \dot{f}(t)$. □

Proposition 5.2.2. *The fluid model of a queueing network operated under LWUU policies is stable.*

Proof. For any regular point t, let $f(t)$ be the function defined above. Suppose that $f(t) > 0$. Then there exists a class (p, k) such that $f_{p,k}(t) = f(t)$. Since $f_{p,k}(t) = \max_{q,l} f_{q,l}(t)$, we have $f_{p,k}(t) \geq f_{p,l}(t)$ for all $l > k$. And because $\beta_{q,l} > 0$ and $\bar{Z}_{p,l}(t) \geq 0$, we have $f_{p,k}(t) \leq f_{p,l}(t)$ for all $l > k$. Thus we have $\bar{Z}_{p,l} = 0$ for all $l > k$. Furthermore we can assume that $\bar{Z}_{p,k}(t) > 0$. Otherwise we can replace class (p, k) with $(p, k-1)$ and still have $f_{p,k-1}(t) = f(t)$. Since $f(t) > 0$, this procedure can keep going and finally a positive buffer will be reached.

Now let $j = \sigma(p, k)$, which is the index of the server that class (p, k) belongs to. Since $\bar{Z}_{p,k}(t) > 0$, we have

$$\sum_{(q,l)\in\mathcal{C}(j)} \dot{\bar{T}}_{q,l}(t) = 1. \tag{5.2.6}$$

Let

$$\mathcal{C}_j^+(t) = \{(q, l) \in \mathcal{C}(j) : f_{q,l}(t) = f(t)\}.$$

By (5.2.5) and (5.2.6) we have

$$\sum_{(q,l)\in\mathcal{C}_j^+(t)} \dot{\bar{T}}_{q,l}(t) = 1.$$

Let $d_{q,l} = \dot{\bar{D}}_{q,l}(t)$. Then we have

$$\sum_{(q,l)\in\mathcal{C}_j^+(t)} m_{q,l} d_{q,l} = 1.$$

Since

$$\dot{f}(t) = \beta_q(\alpha_q - d_{q,l})$$

for all $(q,l) \in \mathcal{C}_j^+(t)$, we have

$$d_{q,l} = \alpha_q - \dot{f}(t)/\beta_q$$

for all $(q,l) \in \mathcal{C}_j^+(t)$. Hence we have

$$\sum_{(q,l)\in\mathcal{C}_j^+(t)} m_{q,l}(\alpha_q - \dot{f}(t)/\beta_q) = 1.$$

So we have

$$\dot{f}(t) = \frac{\sum_{(q,l)\in\mathcal{C}_j^+(t)} m_{q,l}\alpha_q - 1}{\sum_{(q,l)\in\mathcal{C}_j^+(t)} m_{q,l}/\beta_q}.$$

By the definition of $\mathcal{C}_j^+(t)$, we have $\mathcal{C}_j^+(t) \subseteq \mathcal{C}(j)$. Hence

$$\sum_{(q,l)\in\mathcal{C}_j^+(t)} m_{q,l}\alpha_q \leq \sum_{(q,l)\in\mathcal{C}(j)} m_{q,l}\alpha_q < 1.$$

Thus we have

$$\dot{f}(t) < 0.$$

Therefore there exists t_0 such that for all $t > t_0$ we have $f(t) = 0$. □

Theorem 5.2.1. *For a given multi-product deterministic route network, assume the traffic intensity of each station is less than one. Then the multi-product deterministic route network is rate stable under a LWUU policy.*

Proof. The conclusion follows from Theorem 4.3.1 and Proposition 5.2.2. □

5.3 Stability of LWTU policies

In this section, we will show that networks with deterministic routing under LWTU policies are stable. Again, we will prove this by showing that corresponding fluid models are stable. The corresponding fluid model is following.

$$\bar{Z}_{p,k}(t) = \bar{Z}_{p,k}(0) + \bar{A}_{p,k}(t) - \bar{D}_{p,k}(t), p = 1, 2, ...P, k = 1, 2, ...K_p, \quad (5.3.1)$$

$$\bar{A}_{p,k}(t) = \bar{D}_{p,k-1}(t), p = 1, 2, ...P, k = 1, 2, ...K_p, \quad (5.3.2)$$

$$C\bar{T}(t) + \bar{Y}(t) = et, \quad t \geq 0, \quad (5.3.3)$$

$$\bar{Y}_j(t) \text{ increases only when } \bar{U}_j(t) = 0, \ j = 1, ..., J, \quad (5.3.4)$$

$$\dot{\bar{T}}_{p,k}(t) = 0 \quad \text{if} \quad \exists (q, l) \in \mathcal{C}(j) \quad \text{such that} \quad (5.3.5)$$

$$\beta_p(\bar{Z}^+_{p,k}(t) - \bar{Z}^-_{p,k}(t)) < \beta_q(\bar{Z}^+_{q,l}(t) - \bar{Z}^-_{q,l}(t)) \quad \text{and} \quad \bar{Z}_{q,l}(t) > 0, \quad (5.3.6)$$

where $\bar{Z}^+_{p,k}(t) = \sum_{l=1}^{k} \bar{Z}_{p,l}(t)$ for $k = 1, 2, ...K_p$, $\bar{Z}^-_{p,k}(t) = \sum_{l=k+1}^{K} \bar{Z}_{p,l}(t)$ for $k = 1, 2, ...K_p - 1$, and $\bar{Z}^-_{p,K_p}(t) = 0$.

Proposition 5.3.1. *The fluid limits of queueing networks under LWTU policies satisfy the above fluid equations.*

Proof. We only check equation (5.3.6). Verification of other equations is standard. Let $\bar{X}$ be a fluid limit under a LWTU policy. Let $r_n \to \infty$ be a corresponding sequence such that $\bar{\mathbb{X}}^{r_n} \to \bar{\mathbb{X}}$ as $n \to \infty$. For any time t, suppose that there exists class (q, l) such that $\beta_p[\bar{Z}^+_{p,k}(t) - \bar{Z}^-_{p,k}(t)] < \beta_q[\bar{Z}^+_{q,l}(t) - \bar{Z}^-_{q,l}(t)]$. Let

$$\epsilon = \min\{\bar{Z}_{q,l}(t), \quad \beta_q[\bar{Z}^+_{q,l}(t) - \bar{Z}^-_{q,l}(t)] - \beta_p[\bar{Z}^+_{p,k}(t) - \bar{Z}^-_{p,k}(t)]\}.$$

Then by the continuity of $\bar{Z}(t)$, there exists δ such that for all $s \in (t - \delta, t + \delta)$, $\beta_q[\bar{Z}^+_{q,l}(t) - \bar{Z}^-_{q,l}(t)] - \beta_p[\bar{Z}^+_{p,k}(t) - \bar{Z}^-_{p,k}(t)] > \epsilon/2$ and $\bar{Z}_{q,l}(s) > \epsilon/2$. Since $\bar{Z}^{r_n}(\cdot) \to \bar{Z}(\cdot)$

u.o.c. and $r_n \to \infty$ as $n \to \infty$, there exists an N such that for all $n > N$,

$$\sup_{0\le s\le t+\delta} \left|\beta_p[Z^+_{p,k}(r_n s) - Z^-_{p,k}(r_n s)]/r_n - \beta_p[\bar{Z}^+_{p,k}(s) - \bar{Z}^-_{p,k}(s)]\right| < \epsilon/8,$$

$$\sup_{0\le s\le t+\delta} \left|\beta_q[Z^+_{q,l}(r_n s) - Z^-_{q,l}(r_n s)]/r_n - \beta_q[\bar{Z}^+_{q,l}(s) - \bar{Z}^-_{q,l}(s)]\right| < \epsilon/8,$$

and

$$\sup_{0\le s\le t+\delta} |Z^+_{q,l}(r_n s)/r_n - \bar{Z}_{q,l}(s)| < \epsilon/4.$$

Hence for $s \in (t-\delta, t+\delta)$,

$$\beta_q[Z^+_{q,l}(r_n s) - Z^-_{q,l}(r_n s)]/r_n - \beta_p[Z^+_{p,k}(r_n s) - Z^-_{p,k}(r_n s)]/r_n > \epsilon/4,$$

and

$$Z^+_{q,l}(r_n s)/r_n > \epsilon/4.$$

And thus for $s \in (r_n(t-\delta), r_n(t+\delta))$,

$$\begin{aligned}&\beta_q[(Z^+_{q,l}(s) - \xi^+_{q,l}) - (Z^-_{q,l}(s) - \xi^-_{q,l})] - \beta_p[(Z^+_{p,k}(s) - \xi^+_{p,k}) - (Z^-_{p,k}(s) - \xi^-_{p,k})]\\ &> r_n\epsilon/4 - [\beta_q(\xi^+_{q,l} - \xi^-_{q,l}) - \beta_p(\xi^+_{p,k} - \xi^-_{p,k})],\end{aligned}$$

and

$$Z^+_{q,l}(s) > r_n\epsilon/4.$$

So for large enough n, we have

$$\beta_q[(Z^+_{q,l}(s) - \xi^+_{q,l}) - (Z^-_{q,l}(s) - \xi^-_{q,l})] - \beta_p[(Z^+_{p,k}(s) - \xi^+_{p,k}) - (Z^-_{p,k}(s) - \xi^-_{p,k})] > 0,$$

and

$$Z^+_{q,l}(s) > 0.$$

Thus by the definition of the LWTU policy, the server will never work for class (p,k) during $(r_n(t-\delta), r_n(t+\delta))$. So

$$T_{p,k}(r_n(t+\delta)) - T_{p,k}(r_n(t-\delta)) = 0.$$

Divided by r_n and let $n \to \infty$ we have

$$\bar{T}_{p,k}(t+\delta) - \bar{T}_{p,k}(t-\delta) = 0.$$

Hence

$$\dot{\bar{T}}_{p,k}(t) = 0.$$

□

Let $f_{p,k}(t) = \beta_p(\bar{Z}^+_{p,k}(t) - \bar{Z}^-_{p,k}(t))$ and $f(t) = \max_{p=1,2,..P,k=1,2,...k_p} f_{p,k}(t)$. Similar to last section, $f(t)$ is a piecewise linear Lyapunov function of $\bar{Z}(t)$. Again one can check that $f(t)$ is a Lipschitz function of t, and hence is absolutely continuous. As previous section, t is said to be regular if both $\mathbb{X}$ and f are differentiable at t. Whenever a derivative is used at time t, t is assumed to be a regular point.

Lemma 5.3.1. *For any regular time t, we have*

(1) $f(t) \geq 0$.

(2) If $f_{q,l}(t) = f(t) > 0$, there exist $k \leq l$ such that $f_{q,k}(t) = f_{q,l}(t)$ and $\bar{Z}_{q,k}(t) > 0$.

(3) If $f_{q,l}(t) = f(t)$, $\dot{f}_{q,l}(t) = \beta_{q,l}(\alpha_q - d_{q,l})$.

Proof. (1) Let p be any product and k_p be the last step of product p. Then $f(t) \geq f_{p,k_p}(t)$. Since $\bar{Z}^-_{K_p}(t) = 0$, we have $f_{p,k_p}(t) = \beta_p \bar{Z}^+_{K_p}(t) \geq 0$. Hence $f(t) \geq 0$.

(2) Let $f_{q,l}(t) = f(t) > 0$. If $\bar{Z}_{q,l}(t) > 0$, then we are done. Otherwise let k be the largest index such that $k < l$ and $\bar{Z}_{q,k}(t) > 0$. If such k does not exist, then

$\bar{Z}^+_{q,l}(t) = 0$ and hence $f_{q,l}(t) = -\beta_q \bar{Z}^-_{q,l}(t) \leq 0$, which contradicts to the assumption $f_{q,l}(t) = f(t) > 0$. Therefore conclusion (2) is true.

(3)Suppose that $f_{q,l}(t) = f(t)$, then for any $k > l$ $\bar{Z}_{q,k}(t) = 0$. If this is not true, let $k > l$ such that $\bar{Z}_{q,k}(t) > 0$. So

$$\bar{Z}^+_{q,k}(t) \geq \bar{Z}^+_{q,l}(t) + \bar{Z}_{q,k}(t) > \bar{Z}^+_{q,l}(t),$$

and

$$\bar{Z}^-_{q,k}(t) \leq \bar{Z}^-_{q,l}(t) - \bar{Z}_{q,k}(t) < \bar{Z}^-_{q,l}(t).$$

Hence

$$f_{q,k)}(t) = \beta_q(\bar{Z}^+_{q,k}(t) - \bar{Z}^-_{q,k}(t)) > \beta_q(\bar{Z}^+_{q,l}(t) - \bar{Z}^-_{q,l}(t)) = f_{q,l}(t).$$

This contradicts to the fact that $f_{q,l}(t) = f(t)$. So

$$f_{q,l} = \beta_q \bar{Z}^+_{q,l}(t).$$

So conclusion (3) is true. □

Proposition 5.3.2. *The fluid model of queueing network operated under LWTU policies is stable.*

Proof. For any regular point t, let $f(t)$ be the function defined above. Suppose that $f(t) > 0$. Then there exists a class (p, k) such that $f_{p,k}(t) = f(t)$. By the Lemma 5.3.1 we can assume that $\bar{Z}_{p,k}(t) > 0$. Using the part 3 of Lemma 5.3.1, the rest proof follows exactly same as the proof of Proposition 5.2.2. □

Theorem 5.3.1. *For a given multi-product deterministic route network, assume the traffic intensity of each station is less than one. Then the multi-product deterministic route network is rate stable under LWTU policies.*

Proof. The conclusion follows from Theorem 4.3.1 and Proposition 5.3.2. □

Part II

Stabilizing Batch Processing Networks

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 8 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, 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 part 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 in order to achieve 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 or not; (3) which full batch class 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 [24], Bramson [2, 3], Kumar and Kumar [25], Dai and Weiss [14], and Chen and Zhang [8, 9]. In this part, 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 [18]. Two exceptions are Maglaras and Kumar [30] and Kumar and Zhang [26], in which batch processing networks were studied. In [30], a family of discrete review batch policies was shown to maximize the throughput. In [26], a family of fluctuation-smoothing batch policies was shown to maximize the throughput in special networks called reentrant lines by Kumar [23].

In the stability analysis for a standard processing network, the standard tool is to use fluid models as we discuss in part I; see, Rybko and Stolyar [32], Dai [10], Stolyar [35], Dai and Meyn [12], Chen [6], and Bramson [4]. Jennings [22] extended the fluid model tool for processing networks with setups. In this part, as in [26], we also extend the fluid model tool to batch processing networks.

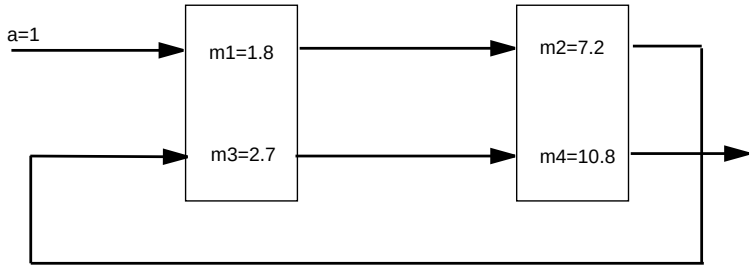

Figure 1: A two-station, four-class batch processing network

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 (6.4.2) in Section 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 (6.4.3) 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.

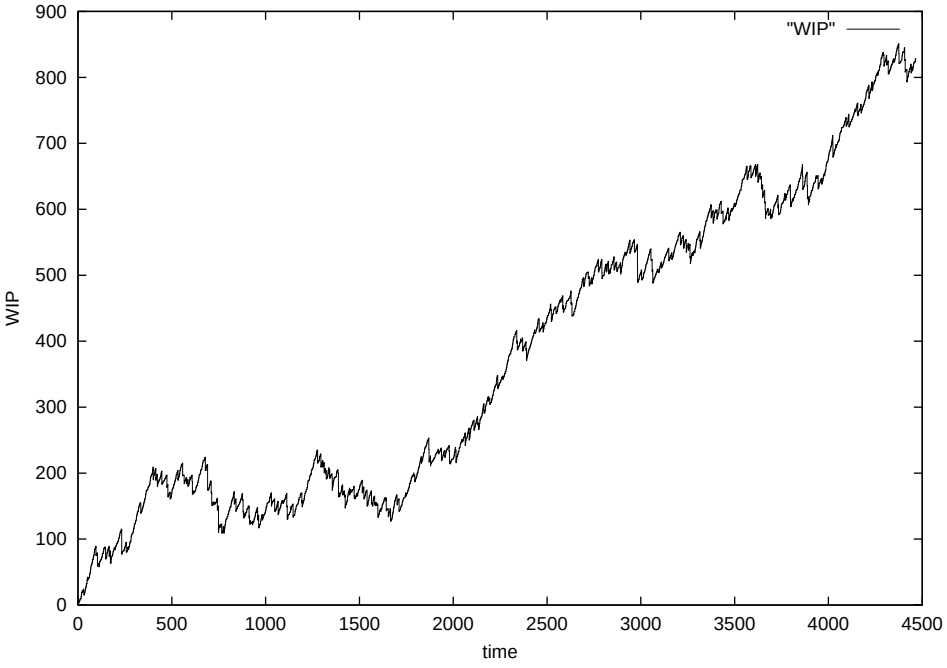

Figure 2: The total number of jobs in system

Since last-buffer-first-serve (LBFS) dispatch policy maximizes the throughput in a standard reentrant line (Dai and Weiss [14] and Kumar and Kumar [25]), 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 table shows the average times in system.

Number of jobs leaving the system	50	500	5000	50000
Average time in system	54.2	208.4	1057.3	6831.6

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 long-run. On the

other hand, the same simulation shows that, after completing 50000 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 Section 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 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. Since this inefficiency phenomenon has been well studied in literature (Kumar-Seidman [24], Lu and Kumar [27], Rybko and Stolyar [32], Bramson [1], and Seidman [33]), we refer readers to these papers for further discussion. (A more recent explanation can be found in Dai and Vande Vate [13] and Hasenbein [20] through virtual and pseudo stations.)

This part is organized as follows. In Chapter 6, 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 this part. In Chapter 7, 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 we introduce fluid limits that are used to justify fluid equations defining a fluid model. In Chapter 8, 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 Chapter 9. These include static buffer priority, first-in–first-out, and generalized round robin policies.

Chapter 6

Open Multi-Class Batch Processing Networks

In this chapter, we first introduce the open multi-class 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.

6.1 The Batch Processing Network

The network under study has J single-server stations and K job classes. Stations are labeled by $j = 1, ..., J$ and classes by $k, \ell = 1, ..., K$. 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 can not 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 $E_k = \{E_k(t), t \geq 0\}$, $V_k = \{V_k(n) : n = 1, 2, ...\}$, and $\Phi^k = \{\Phi^k(n) : n = 1, 2, ...\}$. For each time $t \geq 0$, $E_k(t)$ 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

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

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

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

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

the primitive processes, namely, with probability one,

$$\lim_{t\to\infty} \frac{E_k(t)}{t} = \alpha_k, \quad \lim_{t\to\infty} \frac{V_k(t)}{t} = m_k, \quad \text{and} \quad \lim_{t\to\infty} \frac{\Phi(t)}{t} = P. \tag{6.1.1}$$

The parameter (α, m, P) with $\alpha = (\alpha_1, \ldots, \alpha_K)'$ and $m = (m_1, \ldots, m_K)'$ 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

$$Q = I + P' + (P')^2 + ...$$

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 $S = \{S(t) : t \geq 0\}$ associated with the primitive service process V. For each time $t \geq 0$, $S(t) = (S_1(t), \ldots, S_K(t))'$ with

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

It follows from the strong law of large numbers (6.1.1) that

$$\lim_{t\to\infty} \frac{S_k(t)}{t} = \mu_k, \quad k = 1, \ldots, K, \tag{6.1.2}$$

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 part, 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 Chapter 11 for possible extensions.

6.2 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 $\tilde{V}_k(n) = V_k(n)/B_k$. As a result, the counting process $\tilde{S}$ associated with the primitive service process $\tilde{V}$ is described by $\tilde{S}_k(t) = \max\{n : \tilde{V}_k(n) \leq t\} = S_k(B_k t)$, $k = 1, 2, \ldots, K$, and the strong law of large numbers becomes

$$\lim_{n\to\infty} \frac{\tilde{V}_k(n)}{n} = m_k/B_k \quad \text{and} \quad \lim_{t\to\infty} \frac{\tilde{S}_k(t)}{t} = B_k\mu_k, \quad k = 1, \ldots, K. \tag{6.2.1}$$

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.

Since 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 *dispatch policy*. The alternative term is needed to distinguish the batch policy introduced in the previous section. A major result of this part 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.

6.3 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 part 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 Section 6.4 below.) In formulating our main theorem, Theorem 6.4.1, we need to restrict ourselves to a family of *normal* dispatch policies, whose precise definition will be given in Section 8. Most practical dispatch policies are normal. As an illustration, we prove in Section 9 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.

6.4 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

$$\lambda_l = \alpha_l + \sum_{k=1}^{K} \lambda_k P_{kl}, \text{ for } \ell = 1, 2, ..., K. \tag{6.4.1}$$

In vector form, $\lambda = \alpha + P'\lambda$. Since P is transient, the unique solution to (6.4.1) of λ is given by $\lambda = Q\alpha$. We define the traffic intensity ρ_j for server j (in both networks)

as

$$\rho_j = \sum_{k \in \mathcal{C}(j)} \lambda_k (m_k / B_k), \quad j = 1, \ldots, J, \tag{6.4.2}$$

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

$$\rho_j \leq 1, \quad j = 1, ..., J, \tag{6.4.3}$$

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. 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. Readers are referred to Dai [10] and Bramson [4] for examples and additional discussions of states in standard networks under various policies.

Definition 6.4.1. The batch processing network is *rate stable* if, for each fixed initial state with probability one,

$$\lim_{t \to \infty} \frac{D_k(t)}{t} = \lambda_k, \text{ for } k = 1, ..., K. \tag{6.4.4}$$

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 [16] and the references there). This notion of stability was first introduced for multiclass queueing network settings in Chen [6]. As in a standard network, the usual traffic condition is necessary for rate stability of a batch processing network (Dai [11]). There are other definitions of stability, such as positive Harris recurrence (Dai [10]). The results in this part can be extended to those settings as well.

As mentioned before, the main result of this part 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 Section 8.

Theorem 6.4.1. *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 the theorem will be presented in Section 8. Section 9 is devoted to the applications of Theorem 6.4.1.

Chapter 7

Processing Network and Fluid Model Equations

In this Chapter, 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.

7.1 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 $A = \{A(t), t \geq 0\}$, $D = \{D(t), t \geq 0\}$, $T = \{T(t), t \geq 0\}$, and $Z = \{Z(t), t \geq 0\}$ 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 job count processes, respectively. The components $U = \{U(t), t \geq 0\}$ and $Y = \{Y(t), t \geq 0\}$ 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:

$$A(t) = E(t) + \sum_k \Phi^k(D(t)), \quad t \geq 0, \tag{7.1.1}$$

$$Z(t) = Z(0) + A(t) - D(t), \quad t \geq 0, \tag{7.1.2}$$

$$Z(t) \geq 0, \quad t \geq 0, \tag{7.1.3}$$

$$U(t) = CZ(t), \quad t \geq 0, \tag{7.1.4}$$

$$CT(t) + Y(t) = et, \quad t \geq 0, \tag{7.1.5}$$

$$Y_j(t) \text{ increases only when } Z_k(t) < B_k \text{ for each } k \in \mathcal{C}(j), \quad j = 1, \ldots, J, \tag{7.1.6}$$

$$\text{additional equations associated with the particular batch policy } \tilde{\pi}. \tag{7.1.7}$$

Here C is the constituency matrix defined as

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

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

$$S_k(T_k(t_2)) - S_k(T_k(t_1)) \leq \frac{1}{B_k}(D_k(t_2) - D_k(t_1) + B_k - 1) \tag{7.1.8}$$

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

$$S_k(T_k(t_2)) - S_k(T_k(t_1)) \geq \frac{1}{B_k}(D_k(t_2) - D_k(t_1)). \tag{7.1.9}$$

To check (7.1.8), we note that the left side is the number of class k batches completed in $[t_1, t_2]$. Since 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 maybe smaller than B_k. In any case, the right side of (7.1.8) provides an upper bound on the number of class k batches completed in $[t_1, t_2]$. Thus, inequality (7.1.8) holds. Inequality (7.1.9) can be justified similarly. We call equations (7.1.1)-(7.1.9) 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 non-decreasing. 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 (7.1.8)-(7.1.9) are reduced to

$$\tilde{S}(\tilde{T}(t)) = \tilde{D}(t),\ \text{for all } t \geq 0, \tag{7.1.10}$$

which is well known for standard networks operating under a head-of-line dispatch policy, and equation (7.1.7) is replaced by

additional equations associated with the particular dispatch policy π. (7.1.11)

7.2 Batch and Standard Fluid Models

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

$$\bar{A}(t) = \alpha' t + P'\bar{D}(t), \quad t \geq 0, \tag{7.2.1}$$

$$\bar{Z}(t) = \bar{Z}(0) + \bar{A}(t) - \bar{D}(t), \quad t \geq 0, \tag{7.2.2}$$

$$\bar{Z}(t) \geq 0, \quad t \geq 0, \tag{7.2.3}$$

$$\bar{U}(t) = C\bar{Z}(t), \quad t \geq 0, \tag{7.2.4}$$

$$C\bar{T}(t) + \bar{Y}(t) = et, \quad t \geq 0, \tag{7.2.5}$$

$$\bar{Y}_j(t) \text{ increases only when } \bar{U}_j(t) = 0, \ j = 1, ..., J, \tag{7.2.6}$$

$$\bar{D}_k(t_2) - \bar{D}_k(t_1) \leq B_k\mu_k(\bar{T}_k(t_2) - \bar{T}_k(t_1)), \quad \text{for } 0 \leq t_1 < t_2, \quad k = 1, \ldots, K, \tag{7.2.7}$$

$$\bar{D}_k(t_2) - \bar{D}_k(t_1) = B_k\mu_k(\bar{T}_k(t_2) - \bar{T}_k(t_1)) \text{ if } \bar{U}_j(s) > 0 \ \forall s \in [t_1, t_2], \ 0 \leq t_1 < t_2, \tag{7.2.8}$$

$$\text{additional equations associated with the particular batch policy } \tilde{\pi}. \tag{7.2.9}$$

Equations (7.2.1)-(7.2.9) 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 (7.2.1)-(7.2.9) 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 (7.2.7) and(7.2.8) are replaced by

$$\hat{D}_k(t) = B_k\mu_k\hat{T}_k(t), \text{ for all } t \geq 0, \quad k = 1, \ldots, K, \tag{7.2.10}$$

and (7.2.9) is replaced by

$$\text{additional equations associated with the particular dispatch policy } \pi. \tag{7.2.11}$$

Any process $\hat{\mathbb{X}} = (\hat{A}, \hat{D}, \hat{T}, \hat{U}, \hat{Y}, \hat{Z})$ satisfying equations (7.2.1)-(7.2.6) and equations (7.2.10)-(7.2.11) is called a standard fluid model solution.

Definition 7.2.1. 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 $t \geq 0$.

Weak stability of a standard fluid model can be defined similarly as in Chen [6].

7.3 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

$$\bar{\mathbb{X}}^r(t, \omega) = r^{-1}\mathbb{X}(rt, \omega) \quad t \geq 0. \tag{7.3.1}$$

Note that again for each $r > 0$, $\bar{\mathbb{X}}^r(\cdot, \omega)$ is an element in $\mathbb{D}^{4K+2J}[0, \infty)$. The scaling in (7.3.1) is called the fluid or law-of-large-number scaling.

Definition 7.3.1. 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 (6.1.1) 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 [10] 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 a ω that satisfies (6.1.1). The convergence of $\bar{T}^{r_n}$, together with equation (7.1.9) and condition (6.1.1), implies that $\bar{D}^{r_n}$ converges. This latter convergence, together with equation (7.1.1) and condition (6.1.1), 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 7.3.1. *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 $\mathbb{X}$ be a fluid limit. Equation (7.2.7) follows from (7.1.9). To prove (7.2.8), it is enough to show that for each s such that $\bar{U}_j(s) > 0$ and $\bar{\mathbb{X}}$ is differential at s,

$$\dot{\bar{D}}_k(s) = B_k \mu_k \dot{\bar{T}}_k(s), \tag{7.3.2}$$

where, for a function f, $\bar{f}(s)$ denotes the derivative of f at s. To prove (7.3.2), 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 $\min_{t\in[s-\delta,s+\delta]} \bar{Z}_\ell(t) > 0$. Since $\bar{Z}^{r_n}(\cdot) \to \bar{Z}(\cdot)$ u.o.c., one has that for large enough n,

$$\bar{Z}_\ell(u) \geq B_\ell \quad \text{for} \quad u \in [r_n(s-\delta), r_n(s+\delta)].$$

Now, fix a class k at station j. Since 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, (7.3.2) holds for $t_1 = r_n(s-\delta)$ and $t_2 = r_n(s+\delta)$. Namely,

$$S_k(T_k(r_n(s+\delta))) - S_k(T_k(r_n(s-\delta))) \leq \frac{1}{B_k}(D_k(r_n(s+\delta)) - D_k(r_n(s-\delta)) + B_k - 1).$$

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

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

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

$$\mu_k(\dot{\bar{T}}_k(s) \leq \frac{1}{B_k}\dot{\bar{D}}_k(s). \tag{7.3.3}$$

Equation (7.3.2) now follows from (7.3.3) and (7.2.7). Other fluid model equations can be verified as in Dai [10].

□

Theorem 7.3.1. *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 [6] for the standard processing networks. The proof of our theorem is identical to one for the standard network. See, for example, Dai [11]. □

Chapter 8

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 $t \geq 0$,

$$\hat{A}(t) = \bar{A}(t), \tag{8.0.4}$$

$$\hat{D}(t) = \bar{D}(t), \tag{8.0.5}$$

$$\hat{T}_k(t) = \frac{m_k}{B_k}\hat{D}_k(t), \ k = 1, 2, ..., K, \tag{8.0.6}$$

$$\hat{Y}_j(t) = t - \sum_{k \in \mathcal{C}(j)} \hat{T}_k(t), \quad j = 1, \ldots, J, \tag{8.0.7}$$

$$\hat{U}(t) = \bar{U}(t), \tag{8.0.8}$$

$$\hat{Z}(t) = \bar{Z}(t). \tag{8.0.9}$$

Proposition 8.0.2. *The $\hat{\mathbb{X}}$ constructed from $\bar{\mathbb{X}}$ by (8.0.4)-(8.0.9) satisfies standard fluid model equations (7.2.1)-(7.2.6) and (7.2.10).*

Proof. Since $\hat{Z}(t) = \bar{Z}(t)$, $\hat{A}(t) = \bar{A}(t)$, $\hat{U}(t) = \bar{U}(t)$ and $\hat{D}(t) = \bar{D}(t)$, and since $\bar{Z}(t)$, $\bar{A}(t)$, $\bar{U}(t)$, and $\bar{D}(t)$ satisfy equations (7.2.1)-(7.2.3), $\hat{Z}(t), \hat{A}(t), \hat{U}(t)$, and $\hat{D}(t)$ also satisfy (7.2.1)-(7.2.4). By (8.0.7), equation (7.2.5) is automatically satisfied. Since

$\hat{D}(t)$ is non-decreasing, $\hat{T}(t)$ is also non-decreasing. To show that $\hat{Y}_j$ is non-decreasing, we note that for any $0 \leq t_1 < t_2$,

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

By definitions (8.0.5)-(8.0.6), we have

$$\hat{T}_k(t_2) - \hat{T}_k(t_1) = \frac{m_k}{B_k}(\hat{D}_k(t_2) - \hat{D}_k(t_1)) = \frac{m_k}{B_k}(\bar{D}_k(t_2) - \bar{D}_k(t_1)) \leq \bar{T}_k(t_2) - \bar{T}_k(t_1),$$

where the inequality follows from (7.2.7). Thus we have

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

The last inequality follows from (7.2.5) and the fact that $\bar{Y}_j(\cdot)$ is non-decreasing. Thus, $\hat{Y}_j(t)$ is non-decreasing. 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 (7.2.8), (7.2.5), and (7.2.6) are satisfied for $\bar{\mathbb{X}}$. Thus, equation (7.2.6) is also satisfied for $\hat{\mathbb{X}}$. By (8.0.6), equation (7.2.10) is also true. □

We hope that $\hat{\mathbb{X}}$ also satisfies standard fluid model equation (7.2.11). This, of course, depends on the particular dispatch policy used. As will be shown in the next section, $\hat{\mathbb{X}}$ satisfies (7.2.11) 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 8.0.2. 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 (8.0.4)-(8.0.9) also satisfies (7.2.11).

Proposition 8.0.3. *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 (8.0.4)-(8.0.9). By Proposition 8.0.2 and the definition of a normal policy, $\hat{\mathbb{X}}$ is a standard fluid model solution under dispatch policy π. Since the standard fluid model is weakly stable and $\hat{Z}(0) = 0$, $\hat{Z}(t) = 0$ for $t \geq 0$. But $\bar{Z}(t) = \hat{Z}(t)$ for $t \geq 0$. Hence, we have $\bar{Z}(t) = 0$ for all $t \geq 0$. Thus the batch fluid model under policy $\tilde{\pi}$ is weakly stable. □

With this preparation, the proof of Theorem 6.4.1 follows trivially.

Proof of Theorem 6.4.1. 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 8.0.3, the batch fluid model operating under the induced batch policy $\tilde{\pi}$ is weakly stable. Theorem 6.4.1 then follows from Theorem 7.3.1. □

In the batch and standard fluid models, equations (7.2.9) and (7.2.11) are determined by the batch policy and dispatch policy employed. Examples of these equations will be studied in the next section for FIFO, SBP 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 (7.2.11) often involves $\dot{\bar{T}}(t)$ and (7.2.9) often involves $\dot{\hat{T}}(t)$. The following proposition is often useful to check that a dispatch policy is normal.

Proposition 8.0.4. *Let $\bar{\mathbb{X}}$ be a standard fluid model solution and $\hat{\mathbb{X}}$ be constructed from $\bar{\mathbb{X}}$ by (8.0.4)-(8.0.9). 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 differential 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 (7.2.8) and (8.0.5)-(8.0.6) 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. □

Chapter 9

Examples of Normal Policies

In this Chapter, 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 [2], Chen and Zhang [7, 9], and Dai [11]. Our Theorem 6.4.1 shows that their corresponding induced batch policies preserve the stability property in batch processing networks. Recall that all batch policies are assumed to be non-preemptive, i.e., once a service is started, the server has to finish the service.

9.1 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 few than B_k jobs and class ℓ has at least B_ℓ jobs, then class ℓ has higher priority than class k since class k

is treated as "empty."

Let $H_k = \{\ell : \ell \in \mathcal{C}(j), \pi(\ell) \geq \pi(k)\}$ 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 [14] that the standard fluid model equation (7.2.11) takes the form

$$\dot{\hat{T}}_k^+(t) = 1 \quad \text{for each time } t \text{ such that } \hat{Z}_k^+(t) > 0 \text{ and } \hat{\mathbb{X}} \text{ is differential at } t. \quad (9.1.1)$$

The batch fluid model equation (7.2.9) 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 9.1.1 at the end of this section.

Proposition 9.1.1. *Each fluid limit $\bar{\mathbb{X}}$ of the batch processing network operating under an induced SBP policy satisfies the following equations: for $k = 1, \ldots, K$,*

$$\dot{\bar{T}}_k^+(t) = 1 \quad \textit{for each time } t \textit{ such that } \bar{Z}_k^+(t) > 0 \textit{ and } \bar{\mathbb{X}} \textit{ is differential at time } t. \quad (9.1.2)$$

The main result of this section is the following proposition.

Proposition 9.1.2. *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 (8.0.4)-(8.0.9).

Assume that $\hat{Z}_k^+(t) > 0$ and $\hat{\mathbb{X}}$ is differential at time t. From Proposition 8.0.4 and (9.1.2), we have

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

Thus, $\hat{\mathbb{X}}$ satisfies (9.1.1) and, therefore, is a solution to the standard fluid model. It follows from Definition 8.0.2 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 literature. Under the LBFS policy, classes in later stages have higher priorities. Under the FBFS policy, classs in earlier stages have higher priorities.

Corollary 9.1.1. *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 [14] and Kumar and Kumar [25] that the standard fluid model is weakly stable under FBFS and LBFS dispatch policies. Since these policies are normal, the corollary follows from Theorem 6.4.1. □

Proof of Proposition 9.1.1. 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 (9.1.2), 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 $[0, t]$. Since $\bar{Z}_k^+(t) > 0$, 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.

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

$$\sup_{0\le s\le t+\delta} |Z_k^+(r_n s)/r_n - \bar{Z}_k^+(s)| < \epsilon/2 \quad \text{and} \quad r_n\epsilon/2 \ge |\mathcal{C}(j)| \max_{\ell\in\mathcal{C}(j)} B_\ell,$$

where $Z_k^+(t)$ is the total number of jobs in H_k at time t and $|\mathcal{C}(j)|$ 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 $Z_k^+(s) \ge r_n\epsilon/2$ for $s \in (r_n t - r_n\delta, r_n t + r_n\delta)$. Since $r_n\epsilon/2 \ge |\mathcal{C}(j)| \max_{\ell\in\mathcal{C}(j)} B_\ell$, 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

$$Z_\ell(s) \ge B_\ell. \tag{9.1.3}$$

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 (9.1.3) 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 $r_n t - r_n\delta$ since 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 $Y_k^+(r_n t + r_n\delta) - Y_k^+(r_n t - r_n\delta) \le R^n$ 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 spend $T_\ell(r_n t - r_n\delta)$ 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

$$\frac{1}{r_n}[Y_k^+(r_n t + r_n\delta) - Y_k^+(r_n t - r_n\delta)] \le \max_{\ell\in\mathcal{C}(j)} \frac{V_\ell(S_\ell(T_\ell(r_n(t-\delta))) + 1) - T_\ell(r_n(t-\delta))}{r_n}$$

for $n > N$. Because $\bar{T}^{r_n}(\cdot) \to \bar{T}(\cdot)$, and (6.1.1) and (6.1.2) 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

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

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

9.2 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 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 in order to form a full batch.

For the standard FIFO fluid model, the additional equation (7.2.11) takes the form

$$\hat{D}_k(t + \hat{W}_j(t)) = \hat{A}_k(t), \ k = 1, ..., K \tag{9.2.1}$$

for all $t > 0$, where

$$\hat{W}_j(t) = \sum_{k \in \mathcal{C}(j)} \frac{m_k}{B_k} \hat{Z}_k(t), \ j = 1, \ldots, J. \tag{9.2.2}$$

See, for example, Bramson [2].

For the batch FIFO fluid model, the additional fluid model equation (7.2.9) takes the same form as in (9.2.1) and (9.2.2). This is the content of our next proposition.

Proposition 9.2.1. *Let $\bar{\mathbb{X}}$ be a fluid limit of the batch processing network operating under the FIFO batch policy. It satisfies the following equations:*

$$\bar{D}_k(t + \bar{W}_j(t)) = \bar{A}_k(t), \ k = 1, ..., K, \tag{9.2.3}$$

for all $t > 0$, where

$$\bar{W}_j(t) = \sum_{k \in \mathcal{C}(j)} \frac{m_k}{B_k} \bar{Z}_k(t), \ j = 1, 2, ..., J. \tag{9.2.4}$$

We delay the proof until the end of this section.

Proposition 9.2.2. *The FIFO dispatch policy is normal.*

Proof. Let $\bar{\mathbb{X}}$ be batch fluid model solution under FIFO batch policy. Namely, $\bar{\mathbb{X}}$ satisfies equations (7.2.1)-(7.2.8) and (9.2.3)-(9.2.4). Let $\hat{\mathbb{X}}$ be a fluid solution constructed from $\bar{\mathbb{X}}$ by (8.0.4)-(8.0.9). One can check that $\hat{\mathbb{X}}$ satisfies equations (9.2.1)-(9.2.2). Thus, by Proposition 8.0.2, $\hat{\mathbb{X}}$ is a standard fluid model solution under FIFO dispatch policy. Therefore, 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 9.2.1. *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 [2] that the standard FIFO fluid model of Kelly type is weakly stable under the usual traffic condition. Since the FIFO dispatch policy is normal, the corollary follows from Theorem 6.4.1. □

Proof of Proposition 9.2.1. For the standard FIFO processing network,

$$D_k(t + W_j(t)) = Z_k(0) + A_k(t), \quad k = 1, \ldots, K, \tag{9.2.5}$$

where $W_j(t)$ is the (immediate) workload at station j at time t, from which standard fluid model equations (9.2.1) and (9.2.2) are derived. (See, Harrison and Nguyen [19] and Bramson [2].)

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 (9.2.5). 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 non-full 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

$$Z_k(0) + A_k(t) - B_k < D_k(t + W_j(t)) \text{ for } t \geq 0, \quad k = 1, \ldots, K. \tag{9.2.6}$$

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

$$D_k(t + W_j(t) - \tau_j(t)) < Z_k(0) + A_k(t) + B_k, \quad k = 1, \ldots, K. \tag{9.2.7}$$

To check (9.2.7), 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 (9.2.7).

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 (9.2.3) and (9.2.4), because of (9.2.6) and (9.2.7) it suffices to show that

$$\bar{W}_j^{r_n}(\cdot) \to \sum_{k \mathcal{C}(j)} \frac{m_k}{B_k} \bar{Z}_k(\cdot) \tag{9.2.8}$$

and

$$\bar{\tau}_j^{r_n}(\cdot) \to 0, \tag{9.2.9}$$

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

Let time $t \geq 0$ 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

$$W_j(t) = \sum_{k \in \mathcal{C}(j)} V_k \left(S_k(T_k(t)) + F_k(t) \right) - t + Y_j(t). \tag{9.2.10}$$

So

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

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. Since $\bar{Z}_k^{r_n}(\cdot) \to \bar{Z}_k(\cdot)$ and $\delta_k(t) \leq B_k$ for all $t \geq 0$, 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 $t \geq 0$, we have

$$\bar{V}_k^{r_n} \left(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}^{r_n}(t) \right) \to \bar{T}_k(t) + m_k \bar{Z}_k(t)/B_k, \quad \text{u.o.c.} \tag{9.2.11}$$

Hence,

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

By (7.2.5), we have $\sum_{k\in\mathcal{C}(j)} \bar{T}_k(t) + t - \bar{Y}_j(t) = 0$. Thus,

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

proving (9.2.8).

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

$$\tau_j(t) \le \sum_{k\in\mathcal{C}(j)} \left[V_k\left(S_k(T_k(t)) + F_k(t)\right) - V_k\left(S_k(T_k(t)) + F_k(t) - 1\right)\right].$$

So

$$\bar{\tau}_j^{r_n}(t) \le \sum_{k\in\mathcal{C}(j)} [\bar{V}_k^{r_n}(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}_k^{r_n}(t)) - \bar{V}_k^{r_n}(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}_k^{r_n}(t) - 1)]. \quad (9.2.12)$$

As in (9.2.11), we have

$$\bar{V}_k^{r_n}\left(\bar{S}_k^{r_n}(\bar{T}_k^{r_n}(t)) + \bar{F}^{r_n}(t) - 1\right) \to \bar{T}_k(t) + m_k\bar{Z}_k(t)/B_k, \quad \text{u.o.c.} \quad (9.2.13)$$

as $n \to \infty$. Convergence (9.2.9) follows from (9.2.12), (9.2.11), and (9.2.13). □

9.3 Generalized Round Robin Policies

For a standard network, a generalized round robin (GRR) dispatch policy associated with weight parameter $\beta = (\beta_2, ..., \beta_K)$ is defined as follows. Here each β_k is a positive real number. Recall that $\mathcal{C}(j)$ 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:

$$a_k(n+1) = \lfloor b_k(n) + \beta_k \rfloor, \tag{9.3.1}$$

$$b_k(n+1) = b_k(n) + \beta_k - a_k(n+1), \tag{9.3.2}$$

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, Keshav and Shenker [15] or Parekh and Gallager [31].

The additional standard fluid model equation (7.2.11) takes the form

$$\dot{\hat{T}}_k(t) \geq \frac{\beta_k(m_k/B_k)}{\sum_{\ell \in \mathcal{C}(j)} \beta_\ell(m_\ell/B_\ell)}, \quad k = 1, 2, \cdots, K \tag{9.3.3}$$

for each time t such that $\hat{T}_k(t)$ is differentiable and $\hat{Z}_k(t) > 0$. The intuitive explanation of (9.3.3) is as follows: the average cycle length is at least $\sum_{\ell \in \mathcal{C}(j)} \beta_\ell(m_\ell/B_\ell)$. 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 $\beta_k(m_k/B_k)\Big[\sum_{\ell \in \mathcal{C}(j)} \beta_\ell(m_\ell/B_\ell)\Big]^{-1}$ amount of effort in class k.

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

$$a_k(n+1) = \lfloor b_k(n) + \beta_k/B_k \rfloor, \tag{9.3.4}$$

$$b_k(n+1) = b_k(n) + \beta_k/B_k - a_k(n+1) \tag{9.3.5}$$

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 (7.2.9) takes the same form as (9.3.3). We offer the following proposition.

Proposition 9.3.1. *For each fluid limit $\bar{\mathbb{X}}$ of the batch processing network operating under the induced GRR batch policy, we have*

$$\dot{\bar{T}}_k(t) \geq \frac{(\beta_k/B_k)m_k}{\sum_{\ell \in \mathcal{C}(j)}(\beta_\ell/B_\ell)m_\ell}, \tag{9.3.6}$$

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

The proof is provided at the end of this section.

Proposition 9.3.2. *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 (8.0.9)-(8.0.7). 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 8.0.4 and (9.3.6), we have

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

Thus, $\hat{\mathbb{X}}$ satisfies (9.3.3) and, hence, is a standard fluid model solution. □

Corollary 9.3.1. *Let $\beta = (\beta_2, ..., \beta_K)$ be a vector of positive real numbers. Assume that for each class k,*

$$\frac{(\beta_k/B_k)m_k}{\sum_{l \in \mathcal{C}(j)}(\beta_\ell/B_\ell)m_\ell} \geq \lambda_k m_k/B_k. \tag{9.3.7}$$

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 (9.3.3) and (9.3.7), $\dot{\hat{D}}_k(t) \geq \lambda_k$ for time t such that $\hat{Z}_k(t) > 0$ and $\hat{\mathbb{X}}$ is differential at t. It follows the proof of Proposition x of Bramson [4] that $\hat{Z}(t) = 0$ for $t \geq 0$. Thus, the standard GRR fluid model is weakly stable. Since any GRR dispatch policy is normal, the corollary follows from Theorem 6.4.1. □

Notice that condition (9.3.7) is equivalent to

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

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 9.3.1. 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 (9.3.6) 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

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

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

$$\lim_{n\to\infty} \frac{T_k(r_n t) - T_k(r_n s)}{r_n(t-s)} \geq \frac{(\beta_k/B_k)m_k}{\sum_{l\in\mathcal{C}(j)}(\beta_\ell/B_\ell)m_\ell}. \tag{9.3.8}$$

To study the limit in (9.3.8), 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 9.1.1, one can choose n large enough and δ small enough such that

$$Z_k(t') > \beta_k + B_k \text{ for } t' \in (r_n s, r_n t). \tag{9.3.9}$$

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

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

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 (9.3.9), 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,

$$T_k(r_n t) - T_k(r_n s) \geq V_k(G_k(r_n s) + 1 + \lfloor C_n \beta / B_k \rfloor) - V_k(G_k(r_n s) + 1).$$

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

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

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 $(r_n s, r_n t)$, $G_\ell(r_n t) - G_\ell(r_n s) \leq \lfloor (C_n + 2)\beta_\ell / B_\ell \rfloor$. Hence,

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

Therefore, we have the following inequality:

$$\frac{T_k(r_n t) - T_k(r_n s)}{r_n(t-s)} \geq \frac{C_n^{-1}\left[V_k(G_k(r_n s) + 1 + \lfloor C_n \beta / B_k \rfloor) - V_k(G_k(r_n s) + 1)\right]}{\sum_{\ell \in \mathcal{C}(j)} C_n^{-1}\left[V_\ell\left(G_\ell(r_n s) + \lfloor (C_n + 2)\beta_\ell / B_\ell \rfloor\right) - V_\ell(G_\ell(r_n s))\right]}. \tag{9.3.10}$$

We now claim that

$$\lim_{n \to \infty} C_n^{-1}\left[V_k(G_k(r_n s) + 1 + \lfloor C_n \beta / B_k \rfloor) - V_k(G_k(r_n s) + 1)\right] = \beta_k m_k / B_k. \tag{9.3.11}$$

The claim follows from assumption (6.1.1) and an extension of the strong-law-of-large-numbers (see, for example, Lemma 5.2.1 of Jennings [21]), provided that

$$\limsup_{n \to \infty} G_k(r_n s)/C_n < \infty. \tag{9.3.12}$$

Since $G_k(r_n s) \leq S_k(r_n s)$ and $\lim_{n\to\infty} S_k(r_n s)/r_n \to \mu_k$, (9.3.12) follows from

$$\liminf_{n \to \infty} \frac{C_n}{r_n} > 0, \tag{9.3.13}$$

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

To prove (9.3.13), for each class $\ell \in \mathcal{C}(j)$, 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

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

where the last equality follows from (7.2.8). Moving μ_ℓ to the other side and summing up for $\ell \in \mathcal{C}(j)$, we have

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

where the last equality follows from (7.2.5) and (7.2.6). Hence (9.3.13) is true, and thus (9.3.11) holds.

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

$$\lim_{n\to\infty} C_n^{-1} \left[V_\ell \left(G_\ell(r_n s) + \lfloor (C_n+2)\beta_\ell / B_\ell \rfloor \right) - V_\ell(G_\ell(r_n s)) \right] = \beta_\ell m_\ell / B_\ell. \tag{9.3.14}$$

Inequality (9.3.8), and hence the proposition, follows from (9.3.11), (9.3.14) and (9.3.10). □

Part III

Simulation Studies

Chapter 10

Simulation Studies

In this chapter, we are going to conduct a series of simulation studies to show that 1) the DPPS policies not only can guarantee the maximal throughput, they can also give very good other performance measures such as the average cycle time and variance of cycle time; 2) the algorithm to produce induced batch policies can indeed generate efficient batch policies as long as the original dispatch policies are efficient; 3) a similar algorithm to produce induced setup policies can also generate efficient setup policies as long as the original dispatch policies are efficient. Note that study of setup policies as well as their relationship with corresponding dispatch policies are beyond the scope of this thesis. The algorithm to produce an induced setup policy from a dispatch policy is studied by Jennings [21]. We will introduce the algorithm in Section 10.4.

10.1 A Three-Product-Five-Station Network

In this section, we will introduce a queueing network model which is our basic test bed. In this model, there are three products labeled as Product 1, Product 2 and Product 3, respectively. There are five stations, indexed from 1 to 5. Some stations have only one machine while other stations have more than one machine. When a

Step	Prod 1 proc time (hour)	Prod 2 proc time (hour)	Prod 3 proc time (hour)
1	0.1	0.11	0.1
2	0.75	0.08	0.36
3	0.18	0.1	0.24
4	0.25	0.12	0.2
5	0.5	0.3	0.09
6	0.05	0.08	0.48
7	0.45	0.51	0.05
8	0.15	0.36	0.32
9	0.16	0.05	0.1
10	0.08	0.24	0.12
11	0.15	0.15	0.54
12	0.05	0.36	0.1
13	0.8	0.16	0.28
14	0.05	0.12	0.08
15	0.2	0.05	0.08

Table 1: mean processing time

station has more than one machine, we assume that all machines are homogeneous. In other words, all machines at a station have same average processing rate if they work on the same type of jobs. In this model, station 2 has three machines and station 4 has two machines. All other stations have only one machine. Each product has a route as depicted in Figure 3. Following are the routes for all products.

Product 1 route: $1 \to 2 \to 3 \to 4 \to 5 \to 1 \to 2 \to 3 \to 4 \to 5 \to 1 \to 3 \to 4 \to 5 \to 1$.

Product 2 route: $1 \to 3 \to 4 \to 5 \to 2 \to 1 \to 2 \to 3 \to 5 \to 4 \to 1 \to 2 \to 4 \to 3 \to 5$.

Product 3 route: $1 \to 2 \to 4 \to 5 \to 1 \to 2 \to 5 \to 4 \to 3 \to 1 \to 2 \to 4 \to 3 \to 1 \to 5$.

The processing time of each class is specified in Table 1. In next several subsections, we are going to present simulation studies based on this basic model. First of all, we will compare the performances of different dispatch policies in this model

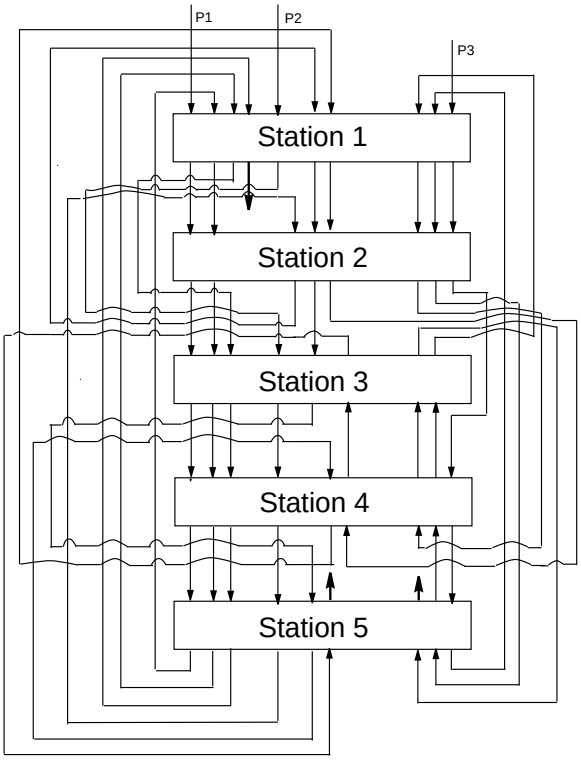

Figure 3: a three-product-five-station network

without batch and setup features. Secondly, we will add batch feature at the second station and modify processing time accordingly. In this modified model, we are going to show that by using induced batch policy algorithm, we can convert an efficient dispatch policy into an efficient batch policy. The purpose of this simulation study is to show that the induced batch policy not only can preserve the stability property of the original dispatch policy, but also can inherit other good features of the original dispatch policy, such as short average cycle time and small variance of cycle time. Finally, besides the batch feature of station 2, we also add setup feature at Stations 1, 3 and 5. By using this model, we show that with both batch operations

and setup delays in the system, the improvement of performance by good policies such as DPPS policies over other commonly used policy such as FIFO combined with Setup Avoidance policy is even more significant.

10.2 Simulation Study of Dispatch Policies

Here, we are going to compare the performance of a DPPS policy, FIFO policy and a LWUU policy on the three-product-five-station network. Before presenting the simulation results, we need to specify the parameters of the DPPS policy and LWUU policy in the simulation. For the DPPS policy, for each server i we select $L_n = \min_{k \in \mathcal{C}(i)} \frac{m_k}{Z_k(t_n)} U_i(t_n)$ in cycle n. Within a cycle, the order that a server picks up a job is according to a rule called quota exhausted round robin which is specified as following. For each station, all classes served by this station are sorted in a list according to pre-fixed order. The server picks the first class in the list and all machines in that station continuously work for this class until 1) there is no more job in the class or 2) the quota of that class is used up. If one of the conditions is satisfied, the station picks the next class in the list. If it reaches the end of the list, it moves to the first class in the list. This procedure ends either there is no more job or there is no more quota in each class, which implies that the cycle ends. Then the station updates the quota and a new cycle begins. The LWUU policy is specified as following. For each product p, $\beta_p = 1$, and for each class (p, j), $\xi_{p,j} = 0$.

As mentioned in Section 10.1, the average processing time of each step is listed in Table 1. We assume that the distributions of processing times are exponential. We also assume that the arrival processes are uncontrollable, and that interarrival

		FIFO		LWUU		DPPS	
Prod	RP	CT	STD	CT	STD	CT	STD
1	1.0	69.9	19.8	46.6	9.7	41.6	14.8
2	2.0	67.6	19.6	39.4	19.7	28.5	9.3
3	1.5	68.2	19.6	57.0	17.5	33.3	10.4

STN	S1	S21	S22	S23	S3	S41	S42	S5
Proc%	93.2	90.4	90.4	90.4	91.5	95.1	95.0	95.6

RP: mean interarrival time; CT: mean cycle time; STD: standard deviation

Table 2: simulation result of case 1

times are also exponential distributed. We compare the performance measures under various traffic intensities by changing each product mean interarrival time. We will start from a high arrival rate for each product and gradually reduce the arrival rate. Table 2 is the simulation results when the mean interarrival time of three products are 1 hour, 2 hours and 1.5 hours, respectively. The first part of Table 2 gives the mean cycle times and cycle time standard deviations. The second part gives the utilization for each machine. As one can see from the table, the DPPS policy has the best performance. The cycle times of all products under the policy DPPS are consistently much less than those under the FIFO policy, as much as 58%.

In the second case, mean interarrival times for Product 1, Product 2, and Produce 3 are changed to 1 hour, 2.2 hours and 1.65 hours, respectively. Again the performance of the DPPS policy is much better than FIFO policy as shown at Table 3.

In the third case, the mean interarrival times for Product 1, Product 2, and Produce 3 are changed to 1.1 hour, 2.7 hours and 2 hours, respectively. Similar to previous two cases, the performance of the DPPS policy is still significantly better

		FIFO		LWUU		DPPS	
Prod	RP	CT	STD	CT	STD	CT	STD
1	1.0	41.3	14.2	29.7	7.6	26.2	10.4
2	2.2	39.7	13.9	21.6	15.8	18.9	6.3
3	1.65	40.3	14.3	43.0	16.8	22.1	7.5

STN	S1	S21	S22	S23	S3	S41	S42	S5
Proc%	89.3	85.9	85.9	85.9	86.7	91.9	91.9	92.5

Table 3: simulation result of case 2

		FIFO		LWUU		DPPS	
Prod	RP	CT	STD	CT	STD	CT	STD
1	1.1	17.7	7.2	17.0	5.5	13.1	5.1
2	2.7	16.5	7.1	11.4	9.1	10.3	3.8
3	2	16.9	7.3	17.6	9.4	11.9	4.3

STN	S1	S21	S22	S23	S3	S41	S42	S5
Proc%	77.7	74.1	74.2	73.8	74.5	80.8	80.8	81.6

Table 4: simulation result of case 3

than FIFO as showed at Table 4. From these three cases, we can see that the DPPS policy has the best performance among three policies.

10.3 Simulation Study of Batch Policies

In this section, the model used in the previous section is modified. Batch operation is added to the second station. The machines at the second station can process a batch of jobs simultaneously. The maximum batch size is 10. The average processing time of a batch is ten times as the processing time of a single job without the batch

feature. From FIFO policy and the DPPS policy used in the previous section, we can get induced batch policies by applying the algorithm discussed in Part II. We call them FIFO batch policy and DPPS batch policy. We compare the performance of the DPPS batch policy with the performance of the FIFO batch policy. For those stations without batch operation, we use corresponding dispatch policy. The mean interarrival times for Product 1, Product 2 and Product 3 are 1.05 hours, 2.2 hours, and 1.65 hours, respectively. Table 5 gives the performance measures of both DPPS batch policy and FIFO batch policy. As we can see, the DPPS batch policy is still significantly better than FIFO batch policy in terms of shorter average cycle time and smaller cycle time standard deviation.

Policy	Prod 1	Prod 2	Prod 3	Total WIP
FIFO	88.5	95.9	93.4	192.7
DPPS	63.5	67.6	71.6	130.6
Improve	28.3%	29.6%	23.4%	32.2%

STN	S1	S21	S22	S23	S3	S41	S42	S5
Proc%	86.9	83.9	84.3	83.7	85.0	89.1	89.0	89.7

Table 5: simulation result of the batch network

10.4 Simulation Study of Batch and Setup Policies

In this section, the model used in the previous section is modified again. The second station's batch operation feature is kept and is the same as specified in the previous section. However, all servers at station 1, station 3 and station 4 have setup delays when they switch from serving one class to another. The mean setup delays of the

machines at the station 1 are 0.5 hour and the setup delays of all machines at station 3 and 5 are 0.3 hours. The procedure of converting a dispatch policy to a setup policy is defined as following. More details can be found in Jennings [21]. Each class is assigned an integer ℓ. The original dispatch policy is used to determine which class to work on. Once a class is selected, the station serves as much as ℓ jobs from the class depending on whether there are ℓ jobs waiting at the station. After finishing those ℓ jobs, the station uses the dispatch policy to choose another class. We will compare the performance of the DPPS policy and the FIFO policy. For the DPPS policy, machines with batch operation will use the induced DPPS batch policy and machines with setup delays will use the induced DPPS setup policy. For the FIFO policy, machines with batch operation will use induced FIFO batch policy, and machines with setup delays will use a policy combining FIFO dispatch policy with setup avoidance. A policy combining FIFO with setup avoid works as following. When a station needs to decide which class to work on next, it uses FIFO dispatch policy to select a class. Then the station simply finishes all jobs in that class before it switches to another class.

Prod	Rate	FIFO CT	DPPS CT	%	FIFO STD	DPPS STD	%
1	0.95	241.1	146.6	39%	48.7	25.6	47%
2	0.408	276.0	157.1	43%	63.8	35.3	45%
3	0.541	275.0	165.8	40%	61.7	32.5	47%

Policy	STN	S1	S21	S22	S23	S3	S41	S42	S5
FIFO	Proc%	83.2	80.7	80.6	80.5	80.4	86.5	86.5	87.4
FIFO	Setup%	16.6	0	0	0	6.2	0	0	6.1
DPPS	Proc%	83.2	80.6	80.5	80.7	80.4	86.6	86.5	87.5
DPPS	Setup%	16.8	0	0	0	10.3	0	0	10.27

Table 6: simulation result of the batch and setup network with smaller arrival rates

We study two cases under different arrival rates. For case one, the mean interarrival times of Product 1, Product 2, and Product 3 are 1.05 hours, 2.45 hours, and 1.85 hours, respectively. The first part of Table 6 displays the mean cycle times and cycle time standard deviations. One can see that the DPPS policy and its induced batch and setup policy have significantly better performance than FIFO and setup avoid policy, on both mean cycle time and cycle time deviation. The second part of Table 6 displays the percentage of time each server spending on processing and setup.

In case two, the arrival rates are increased. Now the the interarrival times for Product 1, Product 2, and Product 3 are 1 hour, 2.2 hours, and 1.65 hours, respectively. In this case, FIFO combining with setup avoidance is no longer stable, meaning that one can not achieve the maximal throughput by using FIFO and setup avoidance, while the DPPS policy and its induced batch and setup policy are still stable. The performance under the DPPS policies is shown in Table 7. One can see that its mean cycle times are increased compared with the lower arrival rates under DPPS. However, they are almost the same as the cycle times of FIFO in the case of the lower arrival rates. In other words, DPPS policies can achieve higher throughput while keeping the cycle time almost the same as those of FIFO policy with smaller throughput.

Prod	Rate	DPPS CT	DPPS STD
1	1	253.3	38.6
2	0.455	275.6	51.6
3	0.606	290.6	50.2

Table 7: simulation result of the batch and setup network with higher arrival rate

Chapter 11

Conclusions and Future Work

We have shown that DPPS policies are stable for all queueing networks with traffic intensity less than one. Also through simulation we demonstrate that DPPS policies have good performances such as short average cycle time. DPPS policies have other advantages. First, to deploy the policy, one does not need to know arrival rate for each class. In practice this information may be not in accurate or even impossible to obtain. Secondly, DPPS policies are very flexible. For each cycle, a DPPS policy only specifies quota for each class. As for the sequence of serving each job, one can apply any heristic algorithm or optimization method to determine. Since a DPPS policy already guarantees stability, throughput rate is always maximized.

In each cycle, DPPS policies calculate quota purely based on the local information. In this sense DPPS policies are distributed dispatch policies. Distributed dispatch policies are simple and easy to implement. However sometime we may want to use more information such as the number of jobs in downstream or upstream buffers as what we did in LWUU policies and LWTU policies. Actually we can modify DPPS policies by changing the way we calculate quota for each class during each cycle. In stead of using local queue length, we can allocate each class's quota proportional to upstream imbalance. The remaining scheme is still same as DPPS policy. One future research direction is to investigate whether such policy is still stable for all queueing

networks with traffic intensity less than one as well as how other performance measures can be improved.

LWUU policies and LWTU policies are proved to be stable for deterministic route queueing networks. One natural question is that how such policies to be extended to probability routing queueing networks and whether such policies are still stable.

For batch networks, we give an algorithm to convert a dispatch policy into a batch policy. We show that if original dispatch policy is stable for a particular standard network and the dispatch policy is a normal policy, then the induced batch policy is also stable for the batch networks. 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 may not be desirable. Suppose that one can choose b_k's with $1 \leq b_k \leq B_k$ such that

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

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.

Another possible extension is that we can slightly modify our definition of "empty" buffer in our policy converting algorithm. For each class k, we can assign a time

window length W_k. Now we define "empty" buffer as following. In the batch network, any class k with fewer than B_k jobs is considered to be "empty". However, if class k has been below B_k longer than W_k and $Z_k(t) > 0$, then class k is again considered as "non-empty". Under new definition of "empty" buffer, sometime non-full batch can still have a chance to be served. It is interesting to investigate under what conditions an induced batch policy converted using such an algorithm can still preserve the stability property of an original dispatch policy.

Bibliography

[1] Bramson, M. Instability of FIFO queueing networks. *Annals of Applied Probability* **4**, 414–431 (1994).

[2] Bramson, M. Convergence to equilibria for fluid models of FIFO queueing networks. *Queueing Systems: Theory and Applications* **22**, 5–45 (1996).

[3] Bramson, M. Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks. *Queueing Systems: Theory and Applications* **23**, 1–26 (1997).

[4] Bramson, M. Stability of two families of queueing networks and a discussion of fluid limits. *Queueing Systems: Theory and Applications* **28**, 7–31 (1998).

[5] Bramson, M. Stablility of earliest-due-date, first-served queueing networks. (2000). Preprint.

[6] Chen, H. Fluid approximations and stability of multiclass queueing networks I: Work-conserving disciplines. *Annals of Applied Probability* **5**, 637–665 (1995).

[7] Chen, H. and Zhang, H. Diffusion approximations for re-entrant lines with a first-buffer-first-served priority discipline. *Queueing Systems: Theory and Applications* **23**, 177–195 (1997).

[8] Chen, H. and Zhang, H. Stability of multiclass queueing networks under FIFO service discipline. *Mathematics of Operations Research* **22**, 691–725 (1997).

[9] Chen, H. and Zhang, H. Stability of multiclass queueing networks under priority service disciplines. *Operations Research* **48**, 26–37 (2000).

[10] Dai, J. G. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Annals of Applied Probability* **5**, 49–77 (1995).

[11] Dai, J. G. Stability of fluid and stochastic processing networks. MaPhySto Miscellanea Publication, No. 9, 1999. Centre for Mathematical Physics and Stochastics.

[12] Dai, J. G. and Meyn, S. P. Stability and convergence of moments for multiclass queueing networks via fluid limit models. *IEEE Transactions on Automatic Control* **40**, 1889–1904 (1995).

[13] Dai, J. G. and VandeVate, J. The stability of two-station multi-type fluid networks. *Operations Research* **48**, 721–744 (2000).

[14] Dai, J. G. and Weiss, G. Stability and instability of fluid models for re-entrant lines. *Mathematics of Operations Research* **21**, 115–134 (1996).

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

[16] El-Taha, M. and Stidham Jr., S. *Sample-Path Analysis of Queueing Systems.* Kluwer, 1999.

[17] Gut, A. *Stopped Random Walks: Limit Theorems and Applications*. Springer, 1988.

[18] Harrison, J. M. Brownian models of queueing networks with heterogeneous customer populations. *Proceedings of the IMA Workshop on Stochastic Differential Systems* (1988). Springer.

[19] Harrison, J. M. and Nguyen, V. Brownian models of multiclass queueing networks: Current status and open problems. *Queueing Systems: Theory and Applications* **13**, 5–40 (1993).

[20] Hasenbein, J. J. Necessary conditions for global stability of multiclass queueing networks. *Operations Research Letters* **21**, 87–94 (1997).

[21] Jennings, O. B. *Multiclass Queueing Networks with Setup Delays: Stability Analysis and Heavy Traffic Approximation*. PhD thesis, School of ISyE, Georgia Institute of Technology, 2000.

[22] Jennings, O. B. On the stability of multiclass queueing networks with setups. Preprint, 2000.

[23] Kumar, P. R. Re-entrant lines. *Queueing Systems: Theory and Applications* **13**, 87–110 (1993).

[24] Kumar, P. R. and Seidman, T. I. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control* **AC-35**, 289–298 (1990).

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

[26] Kumar, S. and Zhang, H. Stability of reentrant lines with batch servers. Preprint, 2000.

[27] Lu, S. H. and Kumar, P. R. Distributed scheduling based on due dates and buffer priorities. *IEEE Transactions on Automatic Control* **36**, 1406–1416 (1991).

[28] Maglaras, C. Discrete-review policies for scheduling stochastic networks: fluid asymptotic optimality. *Annals of Applied Probability* (1998). Submitted.

[29] Maglaras, C. Dynamic scheduling in multiclass queueing networks: stability under discrete-review policies. *Queueing Systems: Theory and Applications* (1998). Submitted.

[30] Maglaras, C. and Kumar, S. Capacity realization in stochastic batch-processing networks using discrete review policies. Preprint, 1999.

[31] Parekh, A. K. and Gallager, R. G. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transaction on Networking* **1**, 344–357 (1993).

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

[33] Seidman, T. I. 'First come, first served' can be unstable! *IEEE Transactions on Automatic Control* **39**, 2166–2171 (1994).

[34] Shu, L., Tom, T., and Donald, C. W. Minimum invertory variability schedule with applications in semiconductor fabrication. *IEEE Transactions on Semiconductor Manufacturing* **9**, 145–149 (1996).

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

Vita

Caiwei Li was born in Yin Xian, Zhejiang Province, China. After completing high school at Yinxian middle school, Caiwei entered Huazhong University of Science and Technology to conduct undergraduate study. After getting his bachelor degree, Caiwei was admitted to Chinese Academy of Sciences in Beijing as a graduate student. There he received a Master of Science degree. After that he went to Georgia Institute of Technology in Atlanta, Georgia to pursue his Ph.d degree.