Robust Scheduling for Queueing Networks

Ramtin Pedarsani

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-58
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-58.html

May 11, 2015

Robust Scheduling for Queueing Networks

by

Ramtin Pedarsani

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering–Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Spring 2015

Robust Scheduling for Queueing Networks

Abstract

Robust Scheduling for Queueing Networks

by

Ramtin Pedarsani

Doctor of Philosophy in Engineering–Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jean Walrand, Chair

Queueing networks are used to model complicated processing environments such as data centers, call centers, transportation networks, health systems, etc. A queueing network consists of multiple interconnected queues with some routing structure, and a set of servers that have different and possibly overlapping capabilities in processing tasks (jobs) of different queues. One of the most important challenges in designing processing systems is to come up with a low-complexity and efficient scheduling policy.

In this thesis, we consider the problem of *robust* scheduling for various types of processing networks. We call a policy robust if it does not depend on system parameters such as arrival and service rates. A major challenge in designing efficient scheduling policies for new large-scale processing networks is the lack of reliable estimates of system parameters; thus, designing a robust scheduling policy is of great practical interest. We develop a novel methodology for designing robust scheduling policies for queueing networks. The key idea of our design is to use the queue-length changes information to learn the right allocation of service resources to different tasks by stochastic gradient projection method. Our scheduling policy is oblivious to the knowledge of arrival rates and service rates of tasks in the network. Further, we propose a new fork-join processing network for scheduling jobs that are represented as directed acyclic graphs. We apply our robust scheduling policy to this fork-join network, and prove rate stability of the network under some mild assumptions.

Next, we consider the stability of *open multiclass queueing networks* under *longest-queue* (LQ) scheduling. LQ scheduling is of great practical interest since (a) it requires only *local* decisions per group of queues; (b) the policy is robust to knowledge of arrival rates, service rates and routing probabilities of the network. Throughput-optimality of LQ scheduling policy for open multiclass queueing network is still an open problem. We resolve the open problem for a special case of multiclass queueing networks with two servers that can each process two queues, and show that LQ is indeed throughput-optimal.

Finally, we consider transportation networks that can be well modeled by queueing networks. We abstractly model a network of signalized intersections regulated by fixed-time controls as a deterministic queueing network with periodic arrival and service rates. This system is characterized by a delay-differential equation. We show that there exists a unique periodic trajectory of queue-lengths, and every trajectory or solution of the system converges to this periodic trajectory, independent of the initial conditions.

To my beloved parents Nahid and Mohammadreza.

Contents

Acknowledgments

First and foremost, I would like to express my sincerest gratitude to my advisor, Prof. Jean Walrand, who has greatly supported me during my Ph.D. studies. Without his guidance and persistent help this thesis would not have been possible. Jean's thorough insight in research and his brilliance in connecting deep mathematical notions with practical engineering ideas have always amazed me. He was not only my advisor, but also a great example of an academic researcher with very positive attitude towards life. I truly appreciate that he let me explore a variety of problems across different areas. One simply could not wish for a better or friendlier advisor.

I would like to thank Prof. Pravin Varaiya, with whom I collaborated on the applications of queueing theory in transportation systems. Chapter 4 of this thesis is the result of collaboration with Pravin. I learned a lot from his great experience, his amazing insight, and his attitude towards research.

I have been very fortunate to collaborate with Prof. Kannan Ramchandran. I truly appreciate his guidance in that project, and his amazing ability to come up with simple tricks to solve complicated problems.

I would like to thank Prof. Yuan Zhong from Columbia University, who was a post-doct in UC Berkeley during my second year of studies. Yuan has been very helpful to me with his knowledge and patience. The second chapter of this thesis is the result of collaboration with him.

I would next like to thank my thesis committee members, Prof. Venkat Anantharam and Prof. David Aldous. Venkat provided great feedback and insight during my research, when we discussed my research problem, or when he attended my talks.

I would also like to thank Mohammad-Ali Maddah-Ali and Urs Niesen, with whom I collaborated as a summer intern at Bell labs. I had great fun doing research with them, and learned many things during my internship.

I would like to thank the members of WIFO/BLISS who have been great friends for me during my studies. Special thanks goes to Kangwook Lee, with whom we had many hours of fruitful discussion. Kangwook has been an amazing collaborator and friend.

Thanks to my friends Babak Damavandi, Khashayar Kotobi, Mostafa Khoshnevisan, Kayvan Samimi, Mohammad Mirhosseini, Amir Hedayat, Nima Anari, Dorsa Sadigh, and Payam Delgosha who greatly supported me in the past 4 years. Special thanks goes to Zahra Amini; without her, my Berkeley years would not have been such a wonderful experience.

Last, but not least, I devote my special thanks to my family. There are no words that can fully express my gratitude to my parents. This thesis is dedicated with love to them.

Chapter 1

Introduction

We are living in a world of large-scale networks such as data centers, call centers, manufacturing lines, transportation networks, etc. As an example, a data center can consist of tens of thousands of servers with overlapping capabilities that can process thousands of jobs. Given the complexity and scale of these new processing networks, researchers and engineers face significant challenges in designing efficient scheduling algorithms. Thus, it is essential to pursue a scientific approach towards this problem.

Queueing network models are widely used for analyzing the performance of large-scale networked systems. These are networks with multiple interconnected queues served by a set of servers. Scheduling and load balancing algorithms for queueing networks need to be designed in such a way that they achieve a large throughput and small delay. Good algorithms can result in substantial savings in investment and energy costs. However, designing provably efficient algorithms is complicated because of the system size, the flexibility of the processing environment, and the variability of the workload.

This thesis mainly focuses on the design and analysis of simple, scalable, and robust scheduling policies for queueing networks. We consider various processing networks that can model different applications such as data centers and transportation systems. We attempt to design novel robust scheduling policies for these networks, and also analyze well-known simple scheduling policies. We call a scheduling policy *robust* if it does not require knowing the parameters of the network. Although the ultimate performance metric for scheduling policies is to minimize response times, here our approach is to design provably throughput-optimal algorithms to avoid intractability of the problem. The focus of this thesis is on the theoretical and methodological aspects; the analytical results should then provide useful guidelines for practical systems.

1.1 Contributions of the Thesis

The main contributions of this thesis are as follows.

- We propose a novel methodology for designing robust policies for processing networks. We call a policy robust if it does not depend on system parameters such as arrival and service rates. Designing robust scheduling policies is of great interest for modern large-scale processing networks that often lack reliable estimates of system parameters. The main novelty of our methodology is to use the stochastic gradient projection method that reacts to queue-length changes in the network in order to find the right allocation of service resources to different tasks that need to be processed.

- We propose a new fork-join processing network that attempts to abstractly model scheduling jobs in data centers and cloud computing. We model the jobs in the network as *directed acyclic graphs* (DAGs). A job in our model is divided to a few smaller tasks, each of them represented by a node of the graph. We assume there are precedence constraints between tasks that are shown by the edges of the graph. We design a virtual queueing network that can fully describe the dynamics of this model, and design a throughput-optimal policy for the virtual queueing network.

- Stability of *longest-queue* scheduling for open multiclass queueing networks is a long-standing open problem. We tackle this problem in this thesis, and show that for a special case of two servers, each of them serving two queues, longest-queue scheduling is indeed maximally stable (or throughput-optimal). We also develop *longest-dominating-queue* policy, which is a new scheduling policy that is provably throughput-optimal for acyclic open multiclass queueing networks.

- Transportation network is an important application of queueing networks. We model a network of signalized intersections regulated by fixed-time (FT) controls as a deterministic queueing network with periodic arrival and service rates. The dynamics of this system is characterized by a set of delay-differential equations. In this system, we show that (a) there exists a unique periodic trajectory of queue-lengths; (b) every trajectory converges to this periodic trajectory, independent of the initial conditions. The periodic trajectory determines every possible performance measure of the entire network, such as delay, travel time, amount of wasted green, etc. Thus, one is able to design optimal FT control for a network considering only this unique periodic trajectory.

1.2 Related Works

Here we give a brief overview of some popular scheduling policies for different types of queueing networks. More detailed literature review is provided in each chapter.

DAG Scheduling. There is a large body of literature on the scheduling of jobs which can be modeled as DAGs. In the systems literature, researchers have long used DAGs to model parallel processing in multiprocessors. In the more recent data center literature, DAGs are used to model data-intensive computations such as MapReduce [22], Dryad [38], Spark [71], etc. In the analytical literature, researchers have considered a dynamic problem where DAGs arrive to the system and get scheduled dynamically over time. The system model for this problem has been mainly called "fork-join" networks. In the studied models, there is no flexibility in the system; that is, tasks of jobs have dedicated servers. Most of the previous works have considered two basic questions regarding these networks: What are the necessary and sufficient conditions for stability of the system? Given stability, what is the steady-state expected completion times of DAGs? See [8, 7, 6, 5] for more details. The flexible DAG scheduling model that we consider in this thesis is new, and different from all the previous related works. Moreover, finding a robust policy was not previously considered for DAG scheduling.

Multiclass Queueing Networks. Open multiclass queueing networks are models of complex processing systems, and are largely studied in the literature [19, 17, 15]. Most of the works have been concerned about finding a simple throughput-optimal policy for these networks. For networks of Kelly type, i.e., when the service rate of each server only depends on the server itself, FIFO scheduling which is a robust policy has been proved to be throughput-optimal [13, 42]. Head-of-Line processor sharing policy has also been shown to be throughput-optimal for open multiclass queueing networks [14]. For general processing networks, the only known class of throughput-optimal scheduling policies are Max-Weight type policies [18], which require knowledge of service rates and routing probabilities in the network. A simple robust policy in open multiclass queueing networks is serving the longest queue per station, or the longest-queue-first scheduling policy [50]. The throughput-optimality of this scheduling policy is still an open problem.

Transportation Networks. In transportation systems, movement of traffic can be modeled as a queueing network. Signal control policies based on queue-length information have been extensively studied in the literature [52, 67, 1, 2]. Max-pressure policy is shown to be provably maximally stable for these networks [67]. Despite the clear advantages of feedback control policies for a network of signalized intersections, in the U.S., 90 percent of traffic signals follow fixed time controls, which operate the signal in a fixed periodic cycle, independent of the traffic state or queue lengths. No analysis of fixed-time control policy has been done for transportation networks.

1.3 Organization

The rest of this thesis is organized as follows:

Chapter 2 In this chapter, we propose a novel methodology for designing robust policies for processing networks. The key idea of our algorithm is to use stochastic gradient projection method that reacts to queue-length changes in the network in order to find the right allocation of service resources to different tasks. Furthermore, we propose a new processing network in which a job workflow is modeled abstractly as a *directed acyclic graph* (DAG), with nodes representing the tasks, and edges representing precedence constraints among the tasks. We show the throughput-optimality of our robust algorithm for this network. The results of this chapter are partly presented in [58, 60, 59].

Chapter 3 In this chapter, we consider the stability of longest-queue scheduling for open multiclass queueing networks. These are open networks with arbitrary routing matrix and several disjoint groups of queues in which at most one queue can be served at a time. Longest-queue scheduling policy is of great practical interest, since it is robust to all network parameters, and it is local as each server only requires the queue-length knowledge of its own group for scheduling. We show that longest-queue scheduling is throughput-optimal for two groups of two queues. We also propose a new scheduling policy called longest-dominating-queue (LDQ) scheduling, which is robust but not local. We prove the throughput-optimality of longest-dominating-queue scheduling when the network topology is acyclic, for an arbitrary number of groups and queues. The results of this chapter are presented in [57].

Chapter 4 In this chapter, we present an analysis of the traffic dynamics in a network of signalized intersections. The intersections are regulated by fixed-time controls, all with the same cycle length or period. We model this transportation network as a queueing network as follows. Vehicles arrive from outside the network at each queue in a deterministic periodic stream. They take a fixed time to travel along each link (from one queue to another queue). Vehicles make turns at intersections in fixed proportions, and eventually leave the network. We show that if the queueing network is stabilizable, starting at any initial condition, the network state converges to a unique periodic orbit. Thus, the effect of initial conditions disappears. The results of this chapter are presented in [53].

Chapter 5 In this chapter, we conclude the thesis by summarizing the results, and presenting some important future research directions.

Chapter 2

Robust Scheduling for Flexible Processing Networks

2.1 Introduction

As modern processing systems (e.g., data centers, hospitals, manufacturing networks) grow in size and sophistication, their infrastructures become more complicated, and a key operational challenge in many such systems is efficient scheduling of processing resources to meet various demands in a timely fashion.

Two common features of modern large-scale processing networks are the following: a) workflows of interdependent tasks, where the completion of one task will produce new tasks to be processed in the system, and b) flexibility of processing resources with overlapping capabilities as well as flexibility of tasks to be processed by multiple servers. To illustrate these two features, consider the scheduling of a simple Mapreduce job [22] of word count of the play "Hamlet" in a data center (see Figure 2.1). *"Mappers"* are assigned the tasks of word count by Act, producing intermediate results, which are then aggregated by the "reducer". In more elaborate workflows, these interdependencies can be more complicated. Also, there is often considerable overlap in the processing capabilities of the data center servers, and flexibility on where tasks can be placed [22]. Similarly, in a healthcare facility such as a hospital, an arriving patient may have a complicated workflow of service/treatment requirements [4], which can also be assigned to doctors and/or nurses with overlapping capabilities.

In this chapter, we mainly focus on *robust* scheduling policies. A scheduling policy decides how server capacities are allocated over time, and it is called robust if the scheduling decisions are based only on past queue sizes, and do not depend on system parameters such as arrival or service rates. Robust scheduling policies are highly desirable in practice, since (a) parameter estimates are often unreliable, and server operating conditions can vary over time, rendering earlier estimates obsolete (see e.g., [41]); and (b) they use only minimal information and adapt to changes in demands and

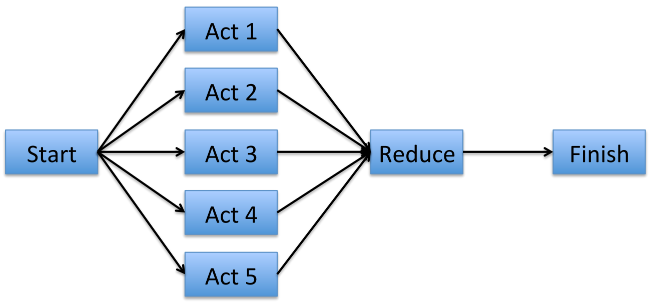

Figure 2.1: Word count of Hamlet in MapReduce

service conditions automatically.

2.1.1 Main Contribution

We provide a theoretical framework of how to design a robust scheduling and capacity allocation policy, and apply this policy to two different processing networks that we propose as follows.

(i) We propose a flexible processing network model, in which arriving jobs are modeled as directed acyclic graphs (DAG), with nodes representing *tasks*, and edges representing *precedence constraints* among tasks. Both servers and tasks are flexible, in the sense that each server is capable of serving a (nonempty) subset of the task types, and each task may be processed by more than one server. The service rate of a server depends on which task type it is serving.

(ii) We consider a flexible queueing network with probabilistic routing structure, where a job goes through processing steps in different queues. Both queues and servers are flexible in the sense that each server is capable of serving a (nonempty) subset of the queues, and tasks at a queue may be served by more than one server. Each queue may have one dedicated exogenous arrival process. Upon service completion, a task may join another queue or leave the network, according to a general routing matrix.

We design a robust scheduling policy for both networks, and prove that when service rates can be written as a product of a server-dependent quantity and a task-dependent quantity, the proposed policy is throughput-optimal[1]. Our policy is based on the simple idea of matching incoming flow rates to their respective service rates, and detecting mismatches using queue size information. If system parameters were known, a so-called static planning problem [33] can be solved to obtain the optimal allocation

[1]We are concerned with *rate stability*. A scheduling policy is *throughput-optimal* if, under this policy, the system is stable whenever there exists some policy under which the system is stable.

of server capacities, which balances flows in the system. Without the knowledge of system parameters, however, the policy updates the allocation of server capacities according to changes in queue sizes. Equivalently, our scheduling policy solves the static planning problem without directly knowing what the parameters are by learning the right capacity allocation observing only the queues sizes. While our policy uses the general idea of stochastic gradient descent (see e.g., [11]), a technique that has been successfully employed in the design of distributed CSMA algorithms for wireless networks with good performance [39], it has major differences compared to the Q-CSMA algorithm: (i) Q-CSMA algorithm is a local adaptive distributed algorithm for maximizing the throughput of wireless networks, but the robust algorithm tries to adaptively find the right share of server's capacities for each task in the network; (ii) Q-CSMA algorithms requires full knowledge of service rates, but our algorithm is robust to the knowledge of task service rates. Finally, we would like to highlight that the proposed methodology is applicable to many different types of queueing networks beyond the two specific networks that are considered here.

2.1.2 Related Works

Scheduling of queueing networks has been studied extensively over several decades. We do not attempt to provide a comprehensive literature review here; instead, we highlight some of the literature, which is closer to the two processing networks we consider.

First, our DAG processing model is closely related to classical fork-join networks (see e.g., [7, 6, 43, 10, 54]). The main difference between the classical models and ours is that we allow tasks to be flexible, whereas tasks are assigned to dedicated servers in classical fork-join networks. To the best of our knowledge, the problem of robust scheduling has not been addressed for the classical fork-join network models. The most basic question regarding these networks concerns the necessary and sufficient conditions for stability, that is, for the existence of the steady-state probability distribution of the underlying Markov process.

Given the stability results, the next natural question is to compute the steady-state expected completion times of DAGs. Few analytical results are available, except for the simplest models (see e.g., [28, 27, 49]). Performance bounds on the stationary expected job completion time have been derived (see e.g., [8, 7, 6, 5]), but for most models, the tightness of these bounds is not known.

An approach that has proved effective in revealing structural properties of complex queueing networks is so-called "heavy-traffic analysis", where the system state is scaled appropriately and system utilization approaches 1. Works on fork-join networks in this direction include [54, 55, 68].

Second, the flexible queueing network model is closely related to the system considered in [3] ([3] also considers setup costs whereas we do not). The policies in [3] make use of arrival and service rates, and their throughput properties are analyzed using fluid models, hence their approach is distinct from ours. We would also like to point out that our network model includes some well-studied queueing systems as special cases. In the case where the queues are not flexible, i.e., each queue has a dedicated server, the system reduces to the open multiclass queueing network (see e.g., [35, 17]). The version of the system with no routing, i.e., when arriving customers leave the network immediately after service completion, is equivalent to the classical flexible parallel server system, considered in e.g., [48].

There is considerable interest in the study of robust scheduling algorithms in the context of parallel server systems. The well-know $Gc\mu$ rule (equivalent to a MaxWeight policy with appropriately chosen weights on queues) has been proved to have good performance properties (including throughput-optimality) (e.g., [48]), and does not depend on arrival rates. [9] studies performance properties of LQF, which is robust to both arrival and service rates, and establishes its throughput-optimality when the so-called activity graph is a tree. [64] established the throughput-optimality of a priority discipline in the many-servers regime, also under the condition that the activity graph is a tree. [24] established the throughput-optimality of LQF under a local pooling condition.

2.2 Scheduling DAGs with Flexible Servers

2.2.1 System Model

We consider a general flexible fork-join processing network, in which jobs are modeled as directed acyclic graphs (DAG). Jobs arrive to the system as a set of tasks, among which there are precedence constraints. Each node of the DAG represents one task type[2], and each (directed) edge of the DAG represents a precedence constraint. More specifically, we consider M classes of jobs, each of them represented by one DAG structure. Let $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$ be the graph representing the job of class m, $1 \le m \le M$, where $\mathcal{V}_m$ denotes the set of nodes of type-m jobs, and $\mathcal{E}_m$ the set of edges of the graph. Let $\mathcal{V} = \cup_{m=1}^{M} \mathcal{V}_m$ and $\mathcal{E} = \cup_{m=1}^{M} \mathcal{E}_m$. We suppose that each $\mathcal{G}_m$ is connected, so that there is an undirected path between any two nodes of $\mathcal{G}_m$. There is no directed cycle in any $\mathcal{G}_m$ by the definition of DAG. Let the number of nodes of job type m be K_m, i.e. $|\mathcal{V}_m| = K_m$. Let the total number of nodes in the network be K. Thus, $\sum_{m=1}^{M} K_m = K$. We index the task types in the system by k, $1 \le k \le K$, starting from job type 1 to M. Thus, task type k belongs to job type $m(k)$ if

$$\sum_{m'=1}^{m(k)-1} K_{m'} < k \le \sum_{m'=1}^{m(k)} K_{m'}.$$

We call node k' a *parent* of node k, if they belong to same job type m, and $(k', k) \in \mathcal{E}_m$. Let $\mathcal{P}_k$ denote the set of parents of node k. In order to start processing a type-k task,

[2]We will often make use of both the concepts of *tasks* and *task types*. To avoid confusion and overburdening terminology, we will use *node* synonymously with *task type* for the rest of the chapter.

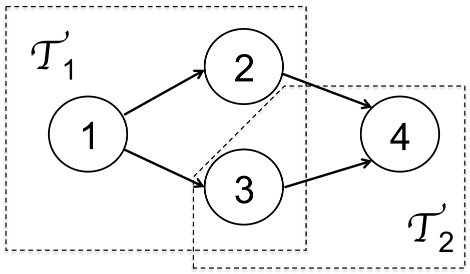

Figure 2.2: A simple DAG

the processing of all tasks of its parents *within the same job* should be completed. Node k is said to be a *root* of DAG type $m(k)$, if $\mathcal{P}_k = \emptyset$. We call k' an *ancestor* of k if they belong to the same DAG, and there exists a directed path of edges from k' to k. Let L_k be the length of the longest path from the root nodes of the DAG, $\mathcal{G}_{m(k)}$, to node k. If k is a root node, then $L_k = 0$.

There are J servers in the processing network. Servers are *flexible* in the sense that each server can serve a non-empty set of nodes. Similarly, nodes/task types are also flexible, so that nodes can be served by a non-empty set of servers. In other words, servers can have overlap of capabilities in processing a node. For each j, we define $\mathcal{T}_j$ to be the set of nodes that server j is capable of serving. Let $T_j = |\mathcal{T}_j|$. For each k, let $\mathcal{S}_k$ be the set of servers that can serve node k, and let $S_k = |\mathcal{S}_k|$. Without loss of generality, we also assume that $T_j, S_k \geq 1$ for all j and k, so that each server can serve at least one node, and each node can be served by at least one server.

Example 1. Figure 2.2 illustrates the DAG of one job type that consists of four nodes $\{1, 2, 3, 4\}$. There are two servers 1 and 2. Server 1 can process tasks of types in the set $\mathcal{T}_1 = \{1, 2, 3\}$ and server 2 can process tasks of types in the set $\mathcal{T}_2 = \{3, 4\}$. When a type-1 task is completed, it "produces" one type-2 task and one type-3 task, both of which have to be completed before the processing of the type-4 task of the same job can start.

We consider the system in discrete time. We assume that the arrival process of type-m jobs is a Bernoulli process with rate λ_m, $0 < \lambda_m < 1$; that is, in each time slot, a new job of type m arrives to the system with probability λ_m, independently over time. We assume that the service times are geometrically distributed and independent of everything else. When server j processes task k, the service completion time has mean μ_{kj}^{-1}. Thus, μ_{kj} can be interpreted as the service rate of node k when processed by server j.

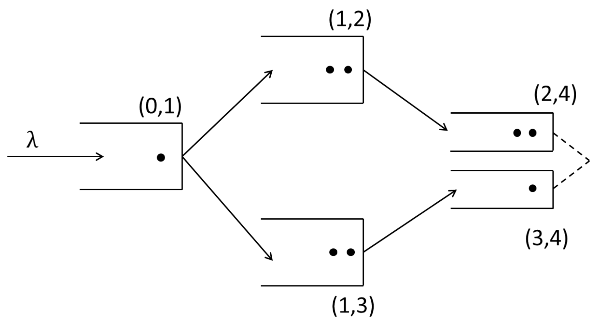

Figure 2.3: Queueing Network of DAG in Figure 2.2

2.2.2 Queueing Network Model for Cooperative Servers

We model our processing system as a queueing network in the following manner. We maintain one virtual queue of processed tasks that are sent from node k' to k for each edge of the DAGs $(k', k) \in \mathcal{E}$. Furthermore, we maintain a virtual queue for the root nodes of the DAGs. Let χ_m be the number of root nodes in the graph of job type m. Then, the queueing network has $\sum_{m=1}^{M}(|\mathcal{E}_m| + \chi_m)$ virtual queues. As an example, consider the DAG of Figure 2.2. The queueing network of this DAG is shown in Figure 2.3. We maintain 5 virtual queues, one for each edge of the graph, and one virtual queue for the root node 1.

Job identities. In our model, jobs and tasks have distinct identities. This is mainly motivated by data center applications as well as health systems. For instance, it is important not to mix up blood samples of different patients in hospital, and to put pictures on the correct webpage in a data center setting. However, in a car manufacturing line, the wheel of the car for example, does not have an identity and can be installed on any car. An approach to avoid mixing tasks of different jobs is to keep track of job identities of all tasks present in the system at all times. However, this requires potentially unbounded memory from the scheduler, since a task could belong to any one of the previously arrived jobs. Furthermore, it is not clear how to best make use of such information for designing policies.

We now explain how a job of identity a is processed in the queueing network of Figure 2.3. When task 1 of job a from queue $(0, 1)$ is processed, tasks 2 and 3 of job a are sent to queues $(1, 2)$ and $(1, 3)$, respectively. When tasks in queues $(1, 2)$ and $(1, 3)$ are processed, their results are sent to queues $(2, 4)$ and $(3, 4)$, respectively. Finally, to process task 4 of job a, one part belonging to job a from queue $(2, 4)$ and one part belonging to job a from $(3, 4)$ are gathered and processed to finish processing job a. We emphasize that tasks are identity-aware in the sense that to complete processing task 4, it is not possible to merge any two parts (belonging to possibly different jobs) from queues $(2, 4)$ and $(3, 4)$.

We describe the synchronization issue in our queueing network model if servers are non-cooperative, through a simple example of DAG shown in Figure 2.2 and its corresponding queueing network shown in Figure 2.3. Suppose that job a arrives shortly before job b to the system. Assume that server 1 is much faster than server 2. The following sequence of events can happen. First, server 1 processes task 1 of job a in queue $(0, 1)$, and then it processes task 1 of job b in queue $(0, 1)$ while server 2 starts processing task 3 of job a in queue $(1, 3)$. Server 1 finishes processing task 2 of job a and the result is sent to queue $(2, 4)$. Then server 1 starts processing task 3 of job b and sends the result to queue $(3, 4)$ before task 3 of job a is fully processed by server 2. Now the head of the line task of queue $(2, 4)$ belongs to job a and the head of the line task of queue $(3, 4)$ belongs to job b. Thus, an identity-oblivious policy may wrongly schedule processing task type 4 by merging the results of jobs a and b from queues $(2, 4)$ and $(3, 4)$. This problem does not arise if the processing of the tasks are FIFO. This is the case if the servers cooperate on the same task, adding their service capacities.

For now, to guarantee synchronization, we consider a simplifying assumption that servers are cooperative. That is, we assume that servers that work on the same task type, cooperate on the same head-of-the-line task, adding their service capacities. We address the non-cooperative case in Section 2.4.

Queue Dynamics. Let $Q_{(k',k)}$ denote the length of the queue corresponding to edge (k', k) and let $Q_{(0,k)}$ denote the length of the queue corresponding to root node k. A task of type k can be processed if and only if $Q_{(k',k)} > 0$ for all $k' \in \mathcal{P}_k$ – this is because servers are cooperative, and tasks are processed in a FIFO manner. Thus, the number of tasks of node k available to be processed is $\min_{k' \in \mathcal{P}_k} Q_{(k',k)}$, if k is not a root node, and $Q_{(0,k)}$, if k is a root node. For example, in Figure 2.3, queue $(2, 4)$ has length 2 and queue $(3, 4)$ has length 1, thus there is one task of type 4 available for processing. When one task of class k is processed, lengths of all queues (k', k) are decreased by 1, where $k' \in \mathcal{P}_k$, and lengths of all queues (k, i) are increased by 1, where $k \in \mathcal{P}_i$. Therefore, the dynamics of the queueing network is as follows. Let $d_k^n \in \{0, 1\}$ be the number of processed tasks of type k at time n, and $a_m^n \in \{0, 1\}$ be the number of jobs of type m that arrives at time n. If k is a root node of the DAG, then

$$Q_{(0,k)}^{n+1} = Q_{(0,k)}^n + a_{m(k)}^n - d_k^n; \tag{2.1}$$

else,

$$Q_{(k',k)}^{n+1} = Q_{(k',k)}^n + d_{k'}^n - d_k^n. \tag{2.2}$$

Let p_{kj} be the fraction of capacity that server j allocates for processing available tasks of class k. We define $p = [p_{kj}]$ to be the *allocation vector*. If server j allocates all its capacity to different tasks, then $\sum_{k \in \mathcal{T}_j} p_{kj} = 1$. Thus, an allocation vector p is called *feasible* if

$$\sum_{k \in \mathcal{T}_j} p_{kj} \leq 1, \ \forall\ 1 \leq j \leq J. \tag{2.3}$$

We interpret the allocation vector at time n, $p^n = [p^n_{kj}]$, as randomized scheduling decisions at time n, in the following manner. First, without loss of generality, the system parameters can always be re-scaled so that $\sum_{j\in\mathcal{S}_k}\mu_{kj}\le 1$ for all k, by speeding up the clock of the system. Now suppose that at time slot n, the allocation vector is p^n. Then, the head-of-the-line task k is served with probability $\sum_{j\in\mathcal{S}_k}\mu_{kj}p^n_{kj}$ in that time slot. Note that $\sum_{j\in\mathcal{S}_k}\mu_{kj}p^n_{kj}\le 1$ by our scaling of the service rates.

2.2.3 The Static Planning Problem

In this subsection, we introduce a linear program (LP) that characterizes the *capacity region* of the network, defined to be the set of all arrival rate vectors λ where there is a scheduling policy under which the queueing network of the system is stable[3]. The *nominal* traffic rate to all nodes of job type m in the network is λ_m. Let $\nu = [\nu_k]\in\mathbb{R}^K_+$ be the set of nominal traffic rate of nodes in the network. Then, $\nu_k = \lambda_m$ if $m(k) = m$, i.e., if $\sum_{m'=1}^{m-1} K_{m'} < k \le \sum_{m'=1}^{m} K_{m'}$. The LP that characterizes the capacity region of the network makes sure that the total service capacity allocated to each node in the network is at least as large as the nominal traffic rate to that node. Thus the LP, known as the *static planning problem* [33], is defined as follows.

$$\begin{aligned}
\text{Minimize}\quad & \rho && (2.4)\\
\text{subject to}\quad & \nu_k \le \sum_{j\in\mathcal{S}_k}\mu_{kj}p_{kj},\ \forall\ 1\le k\le K && \\
& \rho \ge \sum_{k\in\mathcal{T}_j} p_{kj}, \qquad \forall\ 1\le j\le J, && (2.5)\\
& p_{kj} = 0, \qquad\qquad \text{if } k\notin\mathcal{T}_j, && (2.6)\\
& p_{kj}\ge 0. && (2.7)
\end{aligned}$$

Proposition 1. *Let the optimal value of the LP be ρ^*. Then $\rho^*\le 1$ is a necessary and sufficient condition of rate stability of the system.*

The proof of Proposition 1 is provided in Appendix 2.6.1.

By Proposition 1, the *capacity region* Λ of the network is the set of all $\lambda\in\mathbb{R}^M_+$ for which the corresponding optimal solution ρ^* to the LP satisfies $\rho^*\le 1$. More formally,

$$\Lambda \triangleq \left\{\lambda\in\mathbb{R}^M_+ : \exists\ p_{kj}\ge 0 \text{ such that } \sum_{k\in\mathcal{T}_j} p_{kj}\le 1\ \ \forall\ j, \text{ and } \nu_k\le\sum_{j\in\mathcal{S}_k}\mu_{kj}p_{kj}\ \ \forall\ k\right\}. \tag{2.8}$$

[3]As mentioned earlier, the stability condition that we are interested in is rate stability.

2.2.4 Scheduling Policy Robust to Task Service Rates

Consider the following simplifying assumption.

Assumption 1. *For all k and all $j \in \mathcal{S}_k$, service rates μ_{kj} can be factorized to two terms: the service rate of the task type, μ_k, and the service rate or speed of the server, α_j. Thus, $\mu_{kj} = \mu_k \alpha_j$.*

While Assumption 1 appears somewhat restrictive, it covers a variety of important cases. When $\alpha_j = 1$ for all j, the service rates are task dependent. This case models, for example, a data center of servers with the same processing speed (possibly of the same generation and purchased from the same company), but with different software compatibilities, and possibly hosting overlapping sets of data blocks. The case when α_j's are different can model the inherent heterogeneous processing speeds of the servers.

We remark that Assumption 1 is crucial for our proposed robust scheduling algorithm to be throughput-optimal. In Section 2.2.6, we provide an example of a simple network with generic service rates that is unstable under the proposed scheduling policy. In Section 2.4, we relax this assumption by considering generic service rates, and we design a Max-Weight type throughput-optimal policy that is not robust to service rates.

We now propose a scheduling policy with known α_j, which is robust to task service rates μ_k, and prove that it is throughput-optimal. The idea of our scheduling policy is quite simple: it reacts to queue size changes by adjusting the service allocation vector $p = [p_{kj}]$. Since service rates μ_{kj} are factorized to two terms μ_k and α_j, only the sum $p_k \triangleq \sum_{j \in \mathcal{S}_k} \alpha_j p_{kj}$ affects the effective service rate for node k. One can consider p_k as the total capacity that all the servers allocate to node k in a time slot. So, with a slight abuse of notation and terminology, we call $p = [p_k]$ the service allocation vector.

To precisely describe our proposed scheduling algorithm, first we introduce some notation. Let $\mathbf{1}\{Q^n_{(k',k)} > 0\}$ be the indicator that the queue corresponding to edge (k', k) is non-empty at time n. Let $\Delta Q^{n+1}_{(k',k)} = Q^{n+1}_{(k',k)} - Q^n_{(k',k)}$ be the size change of queue (k', k) from time n to $n + 1$. Define E^n_k to be the event that there is a strictly positive number of type-k tasks to be processed at time n. Thus, $E^n_k = \{Q^n_{(0,k)} > 0\}$ if k is a root node, and $E^n_k = \{Q^n_{(k',k)} > 0,\ \forall k' \in \mathcal{P}_k\}$ if k is not a root node. Also let $\mathbf{1}_{E^n_k}$ be the indicator function of event E^n_k.

Let $\mathcal{C} \subseteq \mathbb{R}^K_+$ be the polyhedron of feasible service allocation vector p.

$$\mathcal{C} = \left\{ p \in \mathbb{R}^K : \exists\ p_{kj} \text{ such that } \sum_{j \in \mathcal{S}_k} \alpha_j p_{kj} = p_k \quad \forall k,\ p_{kj} \geq 0 \quad \forall k, j,\ \sum_{k \in \mathcal{T}_j} p_{kj} \leq 1 \quad \forall j \right\}. \tag{2.9}$$

For any K-dimensional vector x, let $[x]_{\mathcal{C}}$ denote the convex projection of x onto $\mathcal{C}$. Finally, let $\{\beta^n\}$ be a positive decreasing sequence with the following properties: (i) $\lim_{n\to\infty} \beta^n = 0$, (ii) $\sum_{n=1}^{\infty} \beta^n = \infty$, (iii) $\sum_{n=1}^{\infty} (\beta^n)^2 < \infty$, and (iv) $\lim_{n\to\infty} \frac{1}{n\beta^n} < \infty$.

As we will see in the sequel, a key step of our algorithm is to find an unbiased estimator of $\lambda - \mu_k p_k$ for all k, based on the current and past queue sizes. Toward this end, for each node k, we first pick a path of queues from a queue corresponding to a root node of the DAG to queue (k', k) for some $k' \in \mathcal{P}_k$. Note that the choice of this path need not be unique. Let $\mathcal{H}_k$ denote the set of queues on this path from a root node to node k. For example, in the DAG of Figure 2.2, for node 4, we can pick the path $\mathcal{H}_4 = \{(0,1),(1,2),(2,4)\}$. Then, we use $\sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}$ as an unbiased estimate of $\lambda - \mu_k p_k$. To illustrate the reason behind this estimate, consider the DAG in Figure 2.3. It is easy to see that

$$\mathbb{E}(\Delta Q^{n+1}_{(0,1)} + \Delta Q^{n+1}_{(1,2)} + \Delta Q^{n+1}_{(2,4)} | Q^n_{(2,4)} > 0, Q^n_{(3,4)} > 0) = \lambda - \mu_4 p_4^n.$$

In general, if $L_k = L-1$ for node k (recall that L_k is the length of the longest path from a root node to k), one picks a path of edges $(i_0,i_1),(i_1,i_2),\ldots,(i_{L-1},i_L)$, such that $i_0 = 0$ and $i_L = k$. Then,

$$\begin{aligned}\mathbb{E}\left[\sum_{l=0}^{L-1} \Delta Q^{n+1}_{(i_l,i_{l+1})} | \mathbf{1}_{E_k^n} = 1\right] &= (\nu_k - \mu_{i_1} p^n_{i_1} \mathbf{1}_{E^n_{i_1}}) + \sum_{l=1}^{L-1} (\mu_{i_l} p^n_{i_l} \mathbf{1}_{E^n_{i_l}} - \mu_{i_{l+1}} p^n_{i_{l+1}} \mathbf{1}_{E^n_{i_{l+1}}}) \\ &= \nu_k - \mu_k p_k^n \mathbf{1}_{E_k^n}. \end{aligned} \tag{2.10}$$

Our scheduling algorithm updates the allocation vector p^n in each time slot n in the following manner.

1. We initialize with an arbitrary feasible p^0.

2. Update the allocation vector p^n as follows.

$$p_k^{n+1} = [p_k^n + \beta^n \mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q^{n+1}_{(i',i)}]_{\mathcal{C}}. \tag{2.11}$$

This completes the description of the algorithm.

We now provide some intuition for the algorithm. As we mentioned, the algorithm tries to find adaptively the capacity allocated to task k, p_k, that balances the nominal arrival rate and departure rate of queues (k', k). The nominal traffic of all the queues of DAG type m is $\nu_{k(m)}$. Thus, the algorithm tries to find $p_k^* = \frac{\nu_k}{\mu_k}$, in which case the nominal service rate of all the queues is $p_k^* \mu_k = \nu_k$. To find an adaptive robust algorithm, we formulate the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{k=1}^{K} (\nu_k - \mu_k p_k)^2 \\ \text{subject to} \quad & p \in \mathcal{C}. \end{aligned} \tag{2.12}$$

Solving (2.12) by the standard gradient descent algorithm, using step size β^n at time n, leads to the update rule

$$p_k^{n+1} = [p_k^n + \beta^n \mu_k(\nu_k - \mu_k p_k^n)]_{\mathcal{C}}. \tag{2.13}$$

To make the update in (2.13) robust, first we consider a "skewed" update

$$p_k^{n+1} = [p_k^n + \beta^n (\nu_k - \mu_k p_k^n)]_{\mathcal{C}}, \tag{2.14}$$

and second, we use the queue-length changes in (2.10) as an unbiased estimator of the term $\nu_k - \mu_k p_k^n$. This results in the update equation in (2.11). Thus, the update in (2.11) becomes robust to knowledge of service rates μ_k and nominal traffic rates ν_k. The algorithm is not robust to knowledge of server rates, α_j, since the convex set $\mathcal{C}$ is dependent on α_j. Thus, the projection requires the knowledge of server speeds α_j.

Let us now provide some remarks on the implementation efficiency of the algorithm. First, the policy is not fully distributed. While the update variables $\mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}^{n+1}$ can be computed locally, the projection $[\cdot]_{\mathcal{C}}$ requires full knowledge of all these local updates. Second, since Euclidean projection on a polyhedron is a quadratic programming problem that can be solved efficiently in polynomial time by optimization algorithms such as the "interior point method" [12], the projection step $[\cdot]_{\mathcal{C}}$ can be implemented efficiently.

The simulation results presented in Subsection 2.2.5 are derived using the described algorithm. For proof purposes, we make a slight modification to the proposed algorithm. First, we assume that a) the nominal arrival rate of all the tasks ν_k is strictly positive, b) there are finitely many servers in the system, and c) all the service rates, μ_{kj}, are finite. Note that assumptions a), b), and c) are trivial assumptions without losing the generality of the network. Then, there exists $\varepsilon_0 > 0$ such that for all k, $\frac{\nu_k}{\mu_k} \geq \varepsilon_0$. We now suppose that ε_0 is known, and consider a variant $\mathcal{C}_{\varepsilon_0}$ of the convex set $\mathcal{C}$, defined to be

$$\mathcal{C}_{\varepsilon_0} = \left\{ p \in \mathbb{R}^K : \exists\ p_{kj} \geq 0 \text{ such that } \sum_{j\in\mathcal{S}_k} \alpha_j p_{kj} = p_k \ \ \forall k,\ p_k \geq \varepsilon_0 \ \ \forall k,\ \sum_{k\in\mathcal{T}_j} p_{kj} \leq 1 \ \ \forall j \right\}. \tag{2.15}$$

Note that $p^* \in \mathcal{C}_{\varepsilon_0}$. We modify the projection to be on the set $\mathcal{C}_{\varepsilon_0}$ every time, so that p^n are now updated as

$$p_k^{n+1} = [p_k^n + \beta^n \mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}^{n+1}]_{\mathcal{C}_{\varepsilon_0}}. \tag{2.16}$$

We remark again that this modification is made only for the purpose of proving Theorem 1, hence for technical reasons.

The main results of this section are the following two theorems.

Theorem 1. *Let $\lambda \in \Lambda$. The allocation vector p^n updated by Equation (2.16) converges to $p^* = [p_k^*]$ almost surely, where $p_k^* = \frac{\nu_k}{\mu_k}$.*

The proof of Theorem 1 is provided in Appendix 2.6.2. Here, we mention three key steps in the proof. First, we show that the non-stochastic gradient projection algorithm with the skewed update (2.14) converges. This is not true in general, but correct here due to the nice property of the objective function in (2.12), which is the sum of separable quadratic terms. Second, we show that the cumulative stochastic noise present in the update due to the error in estimating the correct drift is an L_2-bounded martingale. Thus, by martingale convergence theorem the cumulative noise converges and has a vanishing tail. This shows that after some time the noise becomes negligible. Finally, we prove that the event that all the queues in the network are non-empty happens for a positive fraction of time. Intuitively, this suggests that the algorithm is updating "often enough" to be able to converge. The rigorous justification makes use of Kronecker's lemma [25].

Theorem 2. *Let $\lambda \in \Lambda$. The queueing network representing the DAGs is rate stable under the proposed scheduling policy, i.e.*

$$\lim_{n\to\infty} \frac{Q^n_{(k',k)}}{n} = 0, \ \ a.s., \ \ \forall (k',k).$$

The proof of Theorem 2 is provided in Appendix 2.6.3. While proving the theorem is technically quite involved, the key idea is to use Theorem 1 to prove that the servers allocate enough cumulative capacity to all the tasks in the network, which itself leads to rate stability of the network.

2.2.5 Simulations

In this section, we show the simulation results and discuss the performance of the robust scheduling algorithm. Consider the DAG shown in Figure 2.4. We assume that the system has 1 type of jobs with arrival rate $\lambda = 1/5$. The task service rates are

$$\mu_1 = 1, \mu_2 = 4/3, \mu_3 = 2, \mu_4 = 1/2 \text{ and } \mu_5 = 2/3.$$

The server speeds are $\alpha_1 = 1$ and $\alpha_2 = 1/2$, and $\mathcal{T}_1 = \{1,4,5\}$ and $\mathcal{T}_2 = \{2,3,4\}$. The step size of the algorithm is chosen to be $\beta^n = \frac{1}{n^{0.6}}$ and the initial queue lengths are $[0,0,0]$. From (2.4), one can compute that the capacity region Λ is $\{\lambda \geq 0 : \lambda \leq \frac{6}{23}\}$.

First we demonstrate that our proposed algorithm is throughput-optimal and makes the queues stable. Figure 2.5 illustrates the queue-lengths as a function of time. Moreover, Figure 2.6 shows how vector p^n converges to the flow-balancing values as Theorem 1 states. Figure 2.5 suggests that queues in the network become empty infinitely often, hence are stable. However, the average queue-length is quite large, so the algorithm suffers from bad delay. The reason is that the allocation vector p^n is converging to the

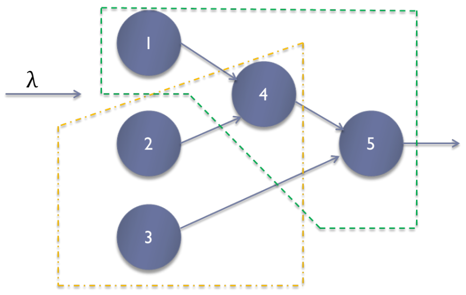

Figure 2.4: DAG of 5 tasks

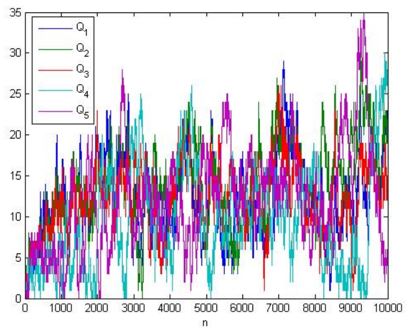

Figure 2.5: Queue-lengths vs. time

value that equalizes the arrival and service rates of all the queues. As an example, if we consider a system with a single-node DAG of arrival rate λ and a single server of service rate μ, the queueing network reduces to the classical M/M/1 queue. Theorem 1 shows that the service capacity that this queue receives, p^n, converges to $\frac{\lambda}{\mu}$. It is known that if the arrival rate of an M/M/1 queue is equal to its service rate, the underlying Markov chain describing the queue-length evolution is null-recurrent, and the queue suffers from large delay. In the following, we propose a modified version of the algorithm that reduces the delays.

Modified scheduling algorithm to improve delays. As discussed in the previous section, the allocation vector p^n converges to the value that just equalizes the arrival rate and the effective service rate that each tasks receives. To improve the delay of the system, one wants to allocate strictly larger service rate to each task than the arrival rate. This is possible only if the arrival vector λ is in the interior of the capacity region. In this case, there exists some $\delta > 0$ and an allocation vector p^* such that $\nu_k \leq -\delta + \mu_k p_k^*$ for all k.

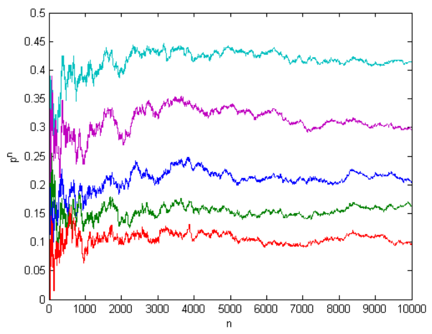

Figure 2.6: Allocation vector p^n vs. time

Thus, assuming that δ is known, we minimize the function

$$V(p) = \sum_{k=1}^{K} (\nu_k + \delta - \mu_k p_k^*)^2,$$

by stochastic gradient. Similarly to the proof of Theorem 1, one can show that p_k^n converges to $\frac{\nu_k+\delta}{\mu_k}$. With this formulation of the optimization problem, the update equation for the new scheduling algorithm is

$$p_k^{n+1} = [p_k^n + \delta + \beta^n \mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}^{n+1}]_{\mathcal{C}},$$

which is similar to (2.11) with an extra δ-slack.

We consider the same setting and network parameters as the previous section, and simulate the modified algorithm using $\delta = 0.03$. Figure 2.7 demonstrates a substantial reduction in the queue-length of queue 4 and the delay performance of the algorithm. Similar plots can be obtained for queue-lengths of other queues, which we omit to avoid redundancy.

Time-varying demand and service and bursty arrivals. In Section 2.1, we mentioned that bursty arrivals as well as time-varying service and arrival rates make the estimation of parameters of the system very difficult and often inaccurate, and used this reason as a main motivation for designing robust scheduling policies. However, the theoretical results are provided for a time-invariant system with memoryless queues. In this section, we investigate the performance of our proposed algorithm in a time-varying system with bursty arrivals.

Consider the same DAG structure of previous simulations and the same server rates. We model the burstiness of demand as follows. At each time slot, a batch of B jobs arrive to the system with probability λ/B. To simulate a time-varying system, we

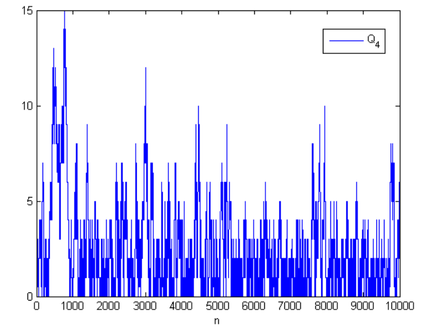

Figure 2.7: Queue-length of queue 4 for the modified algorithm vs. time for $\delta = 0.03$.

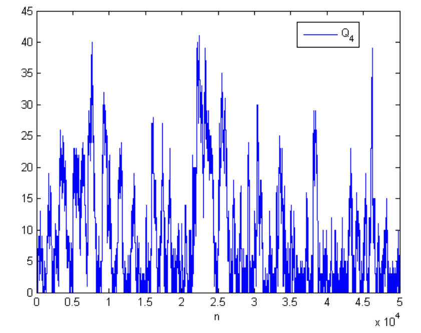

Figure 2.8: Queue-length evolution of queue 4 in the time-varying bursty network.

consider two modes of network parameters. In the first mode, arrival rate is $\lambda = 1/5$, and task service rates are

$$\mu_1 = 1, \mu_2 = 4/3, \mu_3 = 2, \mu_4 = 1/2 \text{ and } \mu_5 = 2/3.$$

In the second mode, arrival rate is $\lambda = 1/6$, and task service rates are

$$\mu_1 = 1/2, \mu_2 = 2, \mu_3 = 1, \mu_4 = 2/5 \text{ and } \mu_5 = 1.$$

Note that the capacity region of mode 2 is $\lambda < 3/14$. We simulate a network that changes mode every T time slots.

Figure 2.8 illustrates the queue-length of the queue 4 versus time when the system has parameters $T = 1000$ and $B = 5$, and $\delta = 0.02$. As one expects, the bursty time-variant system suffers from larger delay. However, as the simulation shows the plotted queue is still stable, and the gradient algorithm is able to track the changes in the network parameters.

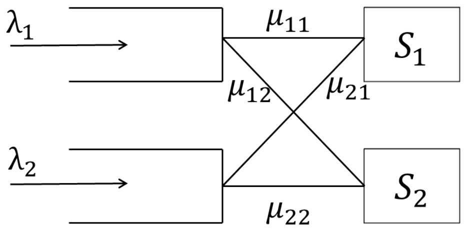

Figure 2.9: The X model

2.2.6 Unstable Network with Generic Service Rates

In this subsection, we first describe a simple modified version of the robust algorithm for the case that service rates are generic.

Define the allocation vector to be $p = [p_{kj}]$. Similar to before, the algorithm tries to minimize $\sum_{k=1}^{K}(\nu_k - \sum_{j=1}^{J} \mu_{kj} p_{kj})^2$ using gradient method. Then, a non-robust update of the allocation vector would be

$$p_{kj}^{n+1} = [p_{kj}^n + \beta^n \mu_{kj}(\nu_k - \sum_{j=1}^{J} \mu_{kj} p_{kj}^n)]_{\mathcal{C}}. \tag{2.17}$$

A robustified version of the update is

$$p_{kj}^{n+1} = [p_{kj}^n + \beta^n \mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}^{n+1}]_{\mathcal{C}}. \tag{2.18}$$

We now demonstrate that the above modified version of the robust algorithm is not throughput-optimal via a simple example. We consider a specific queueing network known in the literature as "X" model [9], shown in Figure 2.9 with generic service rates μ_{kj} that cannot be factorized into 2 factors μ_k and α_j. In our setting, the queueing network is equivalent to having 2 types of DAGs, each of them consisted of a single task with different service characteristics. There are 2 servers in the system. The network parameters are as follows.

$$\lambda_1 = \lambda_2 = 0.3,\ \mu_{11} = \mu_{22} = 1/8,\ \text{and}\ \mu_{12} = \mu_{21} = 3/8.$$

It is easy to check that a corner point of the stability region is $\lambda_1 = 3/8$ and $\lambda_2 = 3/8$, which is achieved by server 1 always working on task-type 2 and server 2 always working on task-type 1. Let $p = [p_{11}, p_{12}, p_{21}, p_{22}]$ be the allocation vector for this example. Then, the update is

$$\begin{bmatrix} p_{1j}^{n+1} \\ p_{2j}^{n+1} \end{bmatrix} = \begin{bmatrix} p_{1j}^n + \beta^n \mathbf{1}_{E_1^n} \Delta Q_1^{n+1} \\ p_{2j}^n + \beta^n \mathbf{1}_{E_2^n} \Delta Q_2^{n+1} \end{bmatrix}_{\mathcal{C}}$$

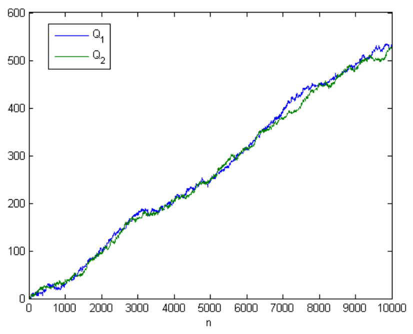

Figure 2.10: Unstable queues in X model

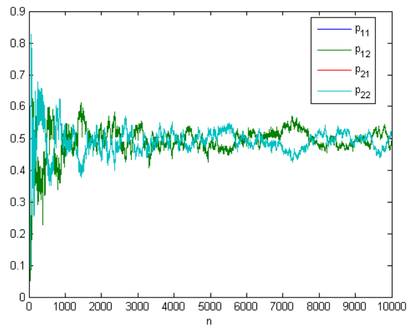

Figure 2.11: Allocation vector p^n converges to $[1/2, 1/2, 1/2, 1/2]$ in X model.

for $j = 1, 2$. We remark that the update of the allocation vector is identical for both servers. Thus, one can predict that the allocation vector converges to $p_{kj} = 1/2$ for all $k \in \{1, 2\}$ and $j \in \{1, 2\}$. The simulation results also show that our scheduling policy is not stable with these network parameters, since the allocation vector converges to $[1/2, 1/2, 1/2, 1/2]$. In this simulation, we set $p^0_{kj} = 0.1$ for all k and j. In general, the reason behind converging to a non-optimal allocation vector for generic μ_{kj} is that the skewed gradient projection (after dropping the term μ_{kj} from (2.17) to (2.18)) does not converge, even without noise.

In general, it would be interesting to find out whether there exists a robust scheduling policy that stabilizes the X model. Due to the underlying symmetry of the problem, it seems plausible that no myopic queue-size policy [4] is throughput-optimal in this example.

[4]These are scheduling policies that are only a function of the current queue sizes of the network.

2.3 Flexible Queueing Network

In this section, we consider a different queueing network model, and show that our robust scheduling policy can also be applied to this network.

2.3.1 Network Model

We consider a flexible queueing network with K queues and J servers, and probabilistic routing. Servers are *flexible* in the sense that each server can serve a (non-empty) set of queues. Similarly, tasks in each queue are *flexible*, so that each queue can be served by a set of servers. Similar to the DAG scheduling model, for each j, let $\mathcal{T}_j$ be the set of queues that server j can serve, let $T_j = |\mathcal{T}_j|$. For each k, let $\mathcal{S}_k$ be the set of servers that can serve queue k, and let $S_k = |\mathcal{S}_k|$. Clearly, $\sum_{j=1}^{J} T_j = \sum_{k=1}^{K} S_k = S$. Without loss of generality, we assume that each server can serve at least one queue, and each queue can be served by at least one server.

We suppose that each queue has a dedicated exogenous arrival process (with rates being possibly zero). For each k, suppose that arrivals to queue k form an independent Bernoulli process with rate $\lambda_k \in [0, 1]$. Thus, in each time slot, there is exactly one arrival to queue k with probability λ_k, and no arrival with probability $1 - \lambda_k$. Let $A_k(t)$ to be the cumulative number of exogenous arrivals to queue k up to time t. The routing structure of the network is described by the matrix $R = [r_{k'k}]_{1 \leq k', k \leq K}$, where $r_{k'k}$ denotes the probability that a task from queue k' joins queue k after service completion. The random routing is i.i.d. over all time slots. We assume that the network is *open*, i.e., all tasks eventually leave the system. This is characterized by the condition that $(I - R^T)$ is invertible, where I is the identity matrix.

Example 2. To clarify the network model, we consider a flexible queueing network shown in Figure 2.12. For concreteness, we can think of this system as a multi-tier application with two flexible servers (the two boxes), and one type of application with three tiers in succession (the three queues). When a task is processed at queue 2, it will join queue 3 with probability r_{23} and it will join queue 1 with probability $r_{21} = 1 - r_{23}$ (that can be thought of as the failure probability in processing queue 2). This network is different from the classical open multiclass queueing networks, in that queue 2 can be served by 2 servers. In this network $\mathcal{T}_1 = \{1, 2\}$ and $\mathcal{T}_2 = \{2, 3\}$, $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{1, 2\}$, and $\mathcal{S}_3 = \{2\}$.

We assume that several servers can work simultaneously on the same task, so that their service capacities can be added. This is equivalent to the case of cooperating servers described in [3]. In each time slot, if a task in queue k is served exclusively by server j, then the task departs from queue k with probability $\mu_{kj} = \mu_j \alpha_j$, where μ_k is the service rate of queue k and α_j is the speed of server j.

The dynamics of the flexible queueing network can be stated as follows. Let Q_k^n be the length of queue k at time n. Let $d_k^n \in \{0, 1\}$ be the number of tasks that depart

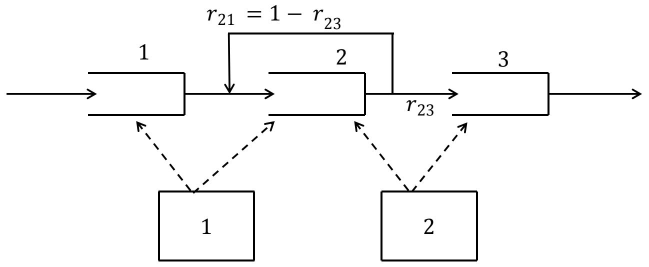

Figure 2.12: Flexible queueing network with 3 queues and 2 servers.

queue k at time n. Let $a_k^n \in \{0,1\}$ be the number of exogenous arrivals to queue k at time n. Finally, let $\mathbf{1}_{k\to k'}^n$ be the indicator that the task departing queue k at time n (if any) is destined to queue k'. Then the queue dynamics is

$$Q_k^{n+1} = Q_k^n + a_k^n + \sum_{k'=1}^{K} d_{k'}^n \mathbf{1}_{k'\to k}^n - d_k^n. \tag{2.19}$$

Note that $\mathbb{E}(a_k^n) = \lambda_k$ and $\mathbb{E}(\mathbf{1}_{k'\to k}^n) = r_{k'k}$.

Similar to the DAG scheduling problem, we define the allocation vector $p = [p_{kj}]$ (of server capacities), and p is called *feasible* if

$$\sum_{k\in\mathcal{T}_j} p_{kj} \le 1, \ \forall\ 1 \le j \le J. \tag{2.20}$$

For each j, p_{kj} can be interpreted as the probability that server j decides to work on queue k. Then, the head-of-the-line task in queue k is served with probability $\sum_j \mu_{kj} p_{kj}$. Note that $\sum_j \mu_{kj} p_{kj} \le 1$ by our scaling of the service rates.

We now introduce the linear program (LP) that characterizes the capacity region of the flexible queueing network. Toward this end, for a given arrival rate vector λ, we first find the *nominal* traffic rates $\nu = [\nu_k]_{1\le k\le K} \in \mathbb{R}^K$, where ν_k is the long-run average total rate at which tasks arrive to queue k. For each k, $\nu_k = \lambda_k + \sum_{i=1}^K \nu_i r_{ik}$. Thus, we can solve ν in terms of R and λ:

$$\nu = (I - R^T)^{-1}\lambda. \tag{2.21}$$

Note that Eq. (2.21) is valid, since by definition, $(I - R^T)$ is invertible. The LP is then defined as follows.

$$\begin{aligned} \text{Minimize} \quad & \rho && (2.22) \\ \text{subject to} \quad & \nu_k \leq \sum_{j \in \mathcal{S}_k} \mu_{kj} p_{kj}, \ \forall\, 1 \leq k \leq K, && (2.23) \\ & \rho \geq \sum_{k \in \mathcal{T}_j} p_{kj}, \qquad \forall\, 1 \leq j \leq J, && (2.24) \\ & p_{kj} = 0, \qquad \text{if } k \notin \mathcal{T}_j, && (2.25) \\ & p_{kj} \geq 0. && (2.26) \end{aligned}$$

Let the optimal value of the LP be ρ^*. Similar to Proposition 1 one can show that $\rho^* \leq 1$ is a necessary and sufficient condition of system stability. Thus, given μ_{kj} and R, the *capacity region* Λ of the network is the set of all $\lambda \in \mathbb{R}_+^K$, so that the corresponding optimal solution ρ^* to the LP satisfies $\rho^* \leq 1$.

2.3.2 Robust Scheduling Policy

In this section, we propose a robust scheduling policy that is provably throughput-optimal when the service rates can be written as $\mu_{kj} = \mu_k \alpha_j$. The policy is robust to arrival and task service rates, but not robust to routing probabilities of the network and servers' speed. The key idea is to use a stochastic gradient projection algorithm to update the service allocation vector p such that all the flows in the network are balanced. We first give the precise description of the algorithm, and state the main theorem. Then, we provide some explanations. We use similar notation as the one used in Section 2.2.

Since service rates can be factorized to a task-dependent rate and a server-dependent rate, only the sum $p_k \triangleq \sum_j \alpha_j p_{kj}$ affects the effective service rate for queue k. So, similar to the DAG scheduling problem, we call $p = [p_k] \in \mathbb{R}^K$ the service allocation vector. Our scheduling algorithm updates the allocation vector p^n in each time slot n in the following manner.

1. We initialize with an arbitrary feasible p^0.

2. Update the allocation vector p^n as follows.

$$p^{n+1} = [p^n + \beta^n \tilde{E}^n (I - R^T)^{-1} \Delta Q^n]_{\mathcal{C}_{\varepsilon_0}}, \tag{2.27}$$

where $\tilde{E}^n$ is a $K \times K$ diagonal matrix such that $\tilde{E}_{kk} = \mathbf{1}_{\{Q_k^n > 0\}}$ and $\mathcal{C}_{\varepsilon_0}$ is given in (2.15).

The main results of this section are the following two theorems.

Theorem 3. *Let $\lambda \in \Lambda$. The allocation vector p^n updated by (2.27) converges to $p^* = [p_k^*]$ almost surely, where $p_k^* = \frac{\nu_k}{\mu_k}$.*

The proof of Theorem 3 is almost identical to the proof of Theorem 1. We avoid repeating the details in the purpose of readability.

Theorem 4. *The flexible queueing network is rate stable under the robust scheduling algorithm specified by the update in* (2.27)*, i.e.*

$$\lim_{n\to\infty} \frac{Q_k^n}{n} = 0, \ \forall k.$$

The proof of Theorem 4 is provided in Appendix 2.6.4.

The intuition for the update (2.27) is as follows. The algorithm tries to adaptively find the allocation vector p^* using a gradient projection method that solves (2.12). To robustify the algorithm to the knowledge of task service rates, we consider the "skewed" updates in (2.14). However, the major difference compared to the DAG scheduling problem is the way we find unbiased estimators of the terms $\nu_k - \mu_k p_k^n$. We use ΔQ^{n+1}, the changes in queue sizes, and routing matrix R, to estimate these terms. It is easy to show that the k^{th} entry of $(I - R^T)^{-1}\Delta Q^{n+1}$ is an unbiased estimator $\nu_k - \mu_k p_k^n$, if $Q_k^n > 0$. Define $M = \text{diag}\{\mu_k\}$. Then,

$$\begin{aligned}
&\mathbb{E}(\tilde{E}^n(I - R^T)^{-1}\Delta Q^{n+1}|Q^n) && (2.28)\\
&= \tilde{E}^n(I - R^T)^{-1}\mathbb{E}(\Delta Q^{n+1}|Q^n) && (2.29)\\
&= \tilde{E}^n(I - R^T)^{-1}(\lambda + R^T M\tilde{E}^n p^n - M\tilde{E}^n p^n) && (2.30)\\
&= \tilde{E}^n\nu - M\tilde{E}^n p^n && (2.31)\\
&= \tilde{E}^n(\nu - Mp^n). && (2.32)
\end{aligned}$$

Note that matrix $\tilde{E}^n$ in update (2.27) ensures that the algorithm updates p_k^n only for queues k that are non-empty, since $\left[(I - R^T)^{-1}\Delta Q^{n+1}\right]_k$ is no longer an unbiased estimator of $\nu_k - \mu_k p_k^n$ when $Q_k^n = 0$.

2.4 Non-cooperative servers

In Section 2.2, we introduced the DAG scheduling problem, and described the synchronization issue when servers are non-cooperative for the queueing network model described in Subsection 2.2.2. In Section 2.2, we considered a simplifying assumption that servers are cooperative to avoid the synchronization problem. In this section, we relax this assumption, and assume that servers are non-cooperative. Instead, we propose a different virtual queueing network that guarantees synchronization in the network.

2.4.1 Virtual Queues for Stages of the Job

We define a stage of a job in the system as the set of tasks belonging to that job that are waiting to be processed. We associate a virtual queue with each possible stage of a

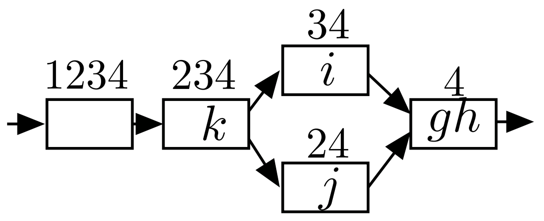

Figure 2.13: Queueing Network of Virtual Queues for the DAG in Figure 2.2

job. This resolves the synchronization issue, since there is no processing activity which requires merging tasks of two or more virtual queues. The following example clarifies how the network of virtual queues is formed.

Example 3. Consider again the DAG shown in Figure 2.2. We consider 5 virtual queues corresponding to 5 possibles stages of the job which are: $\{1, 2, 3, 4\}$, $\{2, 3, 4\}$, $\{2, 4\}$, $\{3, 4\}$, $\{4\}$. The queueing network of these virtual queues is illustrated in Figure 2.13.

Now we formally explain how one forms the network of virtual queues for M types of jobs. Consider M DAGs, $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$, for $1 \leq m \leq M$. We construct M parallel networks of virtual queues as follows. For each non-empty subset $\mathcal{S}_m$ of $\mathcal{V}_m$ consider a virtual queue if and only if for all $i \in \mathcal{S}_m$, all the descendants of i are also in $\mathcal{S}_m$. As an example, subset $\{1, 3\}$ does not correspond to any virtual queue for the DAG in Figure 2.2, since 2 and 4 are descendants of 1, but not in the subset. Let K' be the number of non-empty subsets that satisfy the mentioned condition. Then, we represent the state of M types of DAGs by K' virtual queues. Clearly, there will be no interaction among the virtual queues corresponding to different DAGs.

Remark 1. In general, the number of virtual queues corresponding to different stages of a job with K tasks can grow exponentially in K since each stage denotes a feasible subset of tasks that require processing. This can significantly increase the complexity of scheduling policies that try to maximize the throughput of the network. For practical purposes, it is important to find a queueing network that has low complexity in terms of the number of virtual queues, while it also resolves the synchronization problem.

2.4.2 Queueing Network based on Additional Precedence Constraints

In this subsection, we propose another network of virtual queues that ensure the synchronization of different tasks of one job type. The queueing network is formed by enforcing additional constraints such that the DAG of type m becomes K_m nodes in series. For instance, for the DAG of Figure 2.2, we assume that there is a further constraint that task 3 should proceed task 2. Then, the modified DAG will have 4 nodes in

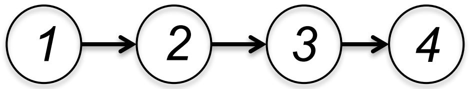

Figure 2.14: Queueing Network of Virtual Queues for the serialized DAG

series as shown in Figure 2.14. The nice property of this "serialized" job is that there are only K_m stages of the job since the stage of each job can be uniquely determined by the next task that requires processing. Therefore, the network of virtual queues has only K queues which decreases substantially the complexity of scheduling policies for the network of virtual queues. However, it is not clear how these extra precedence constraints affect the performance of scheduling policies in terms of both throughput and delay. We provide some preliminary analysis of these effects in Section 2.5.

2.4.3 Throughput-Optimal Policy

In this section, we propose Max-Weight scheduling policy for the network of virtual queues and show that it is throughput-optimal. Note that Max-Weight policy is not robust to the service rates. One can easily extend the results in Section 2.2 to find a robust policy for the virtual queueing networks for the case that $\mu_{kj} = \mu_k \alpha_j$. However, in this section we consider generic service rates, and show that Max-Weight policy is throughput-optimal. We mainly focus on the queueing network mentioned in Subsection 2.4.1. Serializing tasks of different job types is a special case of these queueing networks, though it has substantially fewer number of queues. Therefore, the results provided in this section are also valid for the serialized queueing network mentioned in Subsection 2.4.2.

Now we describe the dynamics of the virtual queues in the network. When server j works on task i in a virtual queue corresponding to subset $\mathcal{S}_m$, the result of the process is sent to the virtual queue corresponding to subset $\mathcal{S}_m \setminus \{i\}$, and the processing rate is μ_{ij}. We call the action of processing task i in virtual queue corresponding to $\mathcal{S}_m$ as a service activity. We denote the collection of different service activities in the network as $\mathcal{A}$. Let $A = |\mathcal{A}|$. Define the collection of activities that server j can perform as $\mathcal{A}_j$.

As an example, consider the DAG of Figure 2.2 and the corresponding queueing network of Figure 2.13. Note that $\mathcal{T}_1 = \{1, 2, 3\}$ and $\mathcal{T}_2 = \{3, 4\}$. Then, there are 8 activities as follows.

1. Task 1 in virtual queue $\{1, 2, 3, 4\}$ is served by server 1 with rate μ_{11}.

2. Task 2 in virtual queue $\{2, 3, 4\}$ is served by server 1 with rate μ_{21}.

3. Task 3 in virtual queue $\{2, 3, 4\}$ is served by server 1 with rate μ_{31}.

4. Task 3 in virtual queue $\{2, 3, 4\}$ is served by server 2 with rate μ_{32}.

5. Task 2 in virtual queue $\{2, 4\}$ is served by server 1 with rate μ_{21}.

6. Task 3 in virtual queue $\{3, 4\}$ is served by server 1 with rate μ_{31}.

7. Task 3 in virtual queue $\{3, 4\}$ is served by server 2 with rate μ_{32}.

8. Task 4 in virtual queue $\{4\}$ is served by server 2 with rate μ_{42}.

Each activity imposes a departure rate from one queue and an arrival rate to another queue. For example, activity 1 leads to the length of virtual queue $\{1, 2, 3, 4\}$ reducing with rate μ_{11}, and virtual queue $\{2, 3, 4\}$ increasing with rate μ_{11}, and so on. For the ease of notation, we index the virtual queues by k', $1 \leq k' \leq K'$. We define a drift matrix $D = [d_{k'a}] \in \mathbb{R}^{K' \times A}$, where $d_{k'a}$ is the rate that virtual queue k' changes if activity a is performed. The drift matrix for the above example is presented in (2.33).

$$D = \begin{pmatrix} -\mu_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu_{11} & -\mu_{21} & -\mu_{31} & -\mu_{32} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu_{31} & \mu_{32} & -\mu_{21} & 0 & 0 & 0 \\ 0 & \mu_{21} & 0 & 0 & 0 & -\mu_{31} & -\mu_{32} & 0 \\ 0 & 0 & 0 & 0 & \mu_{21} & \mu_{31} & \mu_{32} & -\mu_{42} \end{pmatrix}. \tag{2.33}$$

Define length-K' arrival-rate vector $e = [e_{k'}]$ such that $e_{k'} = \lambda_m$ if virtual queue k' corresponds to the first stage of the m-th job type in which no tasks are yet processed, and $e_{k'} = 0$ otherwise. Now we introduce the LP for this virtual queueing network that characterizes the capacity region of this network. Let $k(a)$ be the task that is processed in activity a. Let $\mathcal{A}_j$ be the set of activities done by server j. Let $q = [q_a] \in \mathbb{R}^A$ be the allocation vector. The LP is then defined as follows.

$$\begin{aligned} \text{Minimize} \quad & \rho' && (2.34)\\ \text{subject to} \quad & e + Dq = \mathbf{0} \\ & \rho' \geq \sum_{a \in \mathcal{A}_j} q_a, \qquad \forall\, 1 \leq j \leq J, && (2.35)\\ & q_a \geq 0, && (2.36) \end{aligned}$$

where $\mathbf{0}$ is the vector of zeros, and for length K' vectors x and y, the notation $x < y$ means that $x_{k'} < y_{k'}$ for all k'. The *capacity region* Λ' of the virtual queueing network is the set of all λ (or equivalently the set of corresponding arrival vectors e) for which the solution of the LP satisfies $\rho' \leq 1$.

The arrival-rate vector λ (equivalently the corresponding arrival-rate vector e) is in the interior of capacity region if there exists a feasible capacity allocation vector q^* such that

$$e + Dq^* < \mathbf{0}. \tag{2.37}$$

Note that at a first glance, it is not clear that the LP corresponding to the queueing network mentioned in (2.34) is equivalent to the LP that characterizes the capacity region of the original DAG scheduling problem in (2.4). In the following, we show the equivalence of these two LPs.

It is clear that introducing additional queues cannot increase the capacity region. To see this formally, suppose that there exists a λ and feasible q such that $\rho' \leq 1$. Let $\mathcal{A}'_k$ be the set of activities in which task k is served. Then, by conservation of flow, $\lambda_{m(k)} = \sum_{j:k\in\mathcal{T}_j} \mu_{kj} \sum_{a\in\mathcal{A}'_k\cap\mathcal{A}_j} q_a$ for all k. Now one can choose a feasible allocation vector p for the original DAG scheduling problem such that $p_{kj} = \sum_{a\in\mathcal{A}'_k\cap\mathcal{A}_j} q_a$, and guarantee that $\rho \leq 1$. Thus, $\Lambda' \subseteq \Lambda$, where Λ is the capacity region of the DAG scheduling problem stated in (2.8). To show that $\Lambda \subseteq \Lambda'$, consider some $\lambda \in \Lambda$ and its corresponding allocation vector p. For simplicity, we focus on the example of the DAG shown in Figure 2.2 and the queueing network shown in Figure 2.13. We propose an allocation vector q that can support λ as follows. Fix an ordering on the nodes such that it respects the precedence constraints, for example $(1, 2, 3, 4)$. We design q such that all the flow goes through the path $\{1, 2, 3, 4\}$, $\{2, 3, 4\}$, $\{3, 4\}$, and $\{4\}$. Then, we have

$$\begin{aligned} &q_1 = p_{11}, q_2 = p_{21}, q_3 = 0, q_4 = 0 \\ &q_5 = 0, q_6 = p_{31}, q_7 = p_{32}, q_8 = p_{42}. \end{aligned}$$

It is easy to check that q is feasible (as p is feasible), and also it satisfies flow conservation. Thus, $\Lambda \subseteq \Lambda'$.

Now we give a description of the Max-Weight policy for our virtual queueing system. Given virtual queue-lengths $Q^n_{k'}$ at time n, Max-Weight policy allocates a service vector q that is

$$\arg \max_{q \text{ is feasible}} -(Q^n)^T D q,$$

where $Q^n = [Q^n_{k'}]$ is the vector of queue-lengths at time n. The Max-Weight policy is the choice of q that minimizes the drift or rate of the change of a Lyapunov function $V(Q^n) = \sum_{k'} (Q^n_{k'})^2$. The following example clarifies Max-Weight scheduling in our network.

Example 4. Consider the network shown in Figure 2.13. Assume that the virtual queues are indexed by their corresponding subset of tasks. Then, as an example, at time n, the Max-Weight policy assigns server 1 to task 1 in queue $\{1, 2, 3, 4\}$ if activity 1 has the largest weight among the activities that server 1 can perform. That is,

$$\begin{aligned} &\mu_{11}[Q^n_{1234} - Q^n_{234}] > \mu_{21}[Q^n_{234} - Q^n_{34}], \\ &\mu_{11}[Q^n_{1234} - Q^n_{234}] > \mu_{21}[Q^n_{24} - Q^n_{4}], \\ &\mu_{11}[Q^n_{1234} - Q^n_{234}] > \mu_{31}[Q^n_{234} - Q^n_{24}], \\ &\mu_{11}[Q^n_{1234} - Q^n_{234}] > \mu_{31}[Q^n_{34} - Q^n_{4}]. \end{aligned}$$

The following theorem, similar to Theorem 4 of [18], shows that the Max-Weight policy is rate stable for all the arrival vectors in the capacity region presented in (2.35), and it makes the underlying Markov process of the queue-lengths positive recurrent for all the arrival rate vectors that are in the interior of the capacity region[5].

Theorem 5. *Max-Weight policy for the network of virtual queues described in Subsection 2.4.1 is throughput-optimal.*

Proof. (Sketch) Let us consider the problem in the fluid model. Details of under what conditions and why stability of fluid models implies the stability of stochastic system can be found in [16]. Define the amount of fluid in virtual queue k' as

$$X_{k'}(t) = \lim_{r\to\infty} \frac{Q_{k'}^{\lfloor rt \rfloor}}{r}.$$

Then, the fluid model equations are

$$X(t) = X(0) + et + DT(t),$$

where $X(t) = [X_{k'}(t)]$ is the vector of queue-lengths, $T_a(t)$ is the total time up to t that activity a is served, and $T(t) = [T_a(t)]$ is the vector of total service times of different activities. Further equation incurred by Max-Weight policy is that

$$\dot{T}(t) = \arg \max_{q \text{ is feasible}} -X^T(t)Dq.$$

Now take $V(X(t)) = \frac{1}{2}X^TX$ as the Lyapunov function. The drift of $V(X(t))$ is

$$\begin{aligned} \dot{V}(X(t)) &= X^T(t)(e + D\dot{T}(t)) \\ &= X^T(t)e - \max_q \left(-X^T(t)Dq\right) \\ &\le X^T(t)(e + Dq^*). \end{aligned} \tag{2.38}$$

Now if e is in the interior of the capacity region, we have

$$\dot{V}(X(t)) \le X^T(t)(e + Dq^*) < \mathbf{0}.$$

This proves that the fluid model is stable which implies the positive recurrence of the underlying Markov chain [16]. To prove rate stability, it is sufficient to show that the fluid model is *weakly* stable, that is if $X(0) = \mathbf{0}$, then $X(t) = \mathbf{0}$ for all $t \ge 0$. This is again a direct result of Equation (2.38) since $e + Dq^* \le \mathbf{0}$. ■

Remark 2. The same analysis goes through for the serialized queueing network proposed in Subsection 2.4.2.

[5]Positive recurrence of the Markov chain is a stronger notion of stability compared to the weak notion of rate stability considered in the previous sections.

2.5 Towards Analyzing the Delay Performance

Given the throughput-optimality of the scheduling policies in preceding sections, it is natural to consider and compare their delay performance, e.g., the steady-state expected sojourn time of jobs under the respective policies. However, as one may expect, the exact delay performance of the general model that we consider is very difficult to analyze, if not impossible, as other than for very simple examples, the underlying Markov chain of the system is almost intractable.

In this section, we provide some illustration about the effect of adding precedence constraints among tasks of a job on the delay of processing jobs. Thus, in order to get some intuition about the effect of precedence constraints on delay, we simplify our model and our scheduling policy as follows.

We simplify our model by considering 1 job type that has K identical tasks, and J identical servers. More precisely, we consider jobs that arrive as a Poisson process with rate λ. Each job consists of K tasks, and there are J servers. Each task can be processed by all servers, and requires an independent exponentially distributed service time with mean 1. We compare job scheduling in the following two extreme cases. In the first case, which we call the "parallel" system, $\mathcal{P}$, the K tasks of a job can be processed in parallel; that is, the servers are allowed to work in parallel on two tasks of the same job. We consider a *First-Come-First-Serve* (FCFS) scheduling policy. In the second case, we consider a serial system, $\mathcal{S}$, in which the servers can work only on one task of a job at a time, and they also do so on a FCFS basis. Thus, the system $\mathcal{S}$ imposes a serial execution of the K tasks of each job; that is, a server must complete task 1 before it can start task 2, and so on, similar to the virtual queueing network considered in Subsection 2.4.2.

We define the state of the parallel system to be

$$Z(\mathcal{P},t) = (X(\mathcal{P},t), Y(\mathcal{P},1,t), \dots, Y(\mathcal{P},K-1,t)),$$

where $X(\mathcal{P},t)$ is the number of jobs in the parallel system at time t, and $Y(\mathcal{P},k,t)$ is the number of jobs for which k of the K tasks are already completed at time t. Similarly we define

$$Z(\mathcal{S},t) = (X(\mathcal{S},t), Y(\mathcal{S},1,t), \dots, Y(\mathcal{S},K-1,t)),$$

where $X(\mathcal{S},t)$ is the number of jobs in the serial system at time t, and $Y(\mathcal{S},k,t)$ is the number of jobs for which k of the K tasks are already completed at time t. Note that $(Z(\mathcal{P},t), t \geq 0)$ and $(Z(\mathcal{S},t), t \geq 0)$ are Markov processes.

Clearly, the capacity region of the two systems is the following set of arrival rates: $K\lambda \leq J$. Indeed, work arrives at the systems at rate $K\lambda$ and is performed at rate J when there is enough work to be processed.

Theorem 6. *The Markov process corresponding to the two systems under FCFS scheduling is positive recurrent if $\lambda < J/K$.*

Proof. We use Foster-Lyapunov theorem to show the positive recurrence of the Markov chain. The following analysis holds for both systems. Let $W(t) = KX(t) - \sum_{k=1}^{K-1} kY(k,t)$ denote the number of tasks yet to be processed in the system. Suppose that $\lambda - J/K < -\epsilon$. Then, if $w > J$,

$$\mathbb{E}[W(t+dt) - W(t)|W(t) = w] = K\lambda dt - Jdt < -K\epsilon dt.$$

So the drift of the Lyapunov function $W(t)$ is negative outside the bounded set $W(t) \leq J$, and the Markov chain is positive recurrent. ■

The key result of this section is the following.

Theorem 7. *Let D_S and D_P be the average delays in the two systems. Then,*

$$D_S \leq D_P + \frac{2K-1}{\rho},$$

where $\rho = \frac{K\lambda}{J}$ is the load of the system.

Remark 3. Theorem 7 shows that the delay in the serial system is approximately the same as in the parallel system when the load ρ is not small, and K is not large. This result is of course limited to the case that the servers and tasks are identical, but it suggests that enforcing additional precedence constraints to a DAG scheduling problem does not significantly increase the delay when the load of the network is not small.

Proof. To prove the theorem, we use a coupling argument as follows.

The key observation is that when $X(t) \geq J$, the J servers are all busy in both systems, so that they complete a task at the same rate J. We can then couple the task completion times and the arrivals in the two systems.

Let

$$W(\mathcal{S}, t) = KX(\mathcal{S}, t) - \sum_{k=1}^{K-1} kY(\mathcal{S}, k, t)$$

be the number of tasks to be completed in the serial system and $W(\mathcal{P}, t)$ the corresponding value for the parallel system.

Lemma 1. *It is possible to couple the systems so that*

$$W(\mathcal{S}, t) \leq W(\mathcal{P}, t) + KJ, \forall t \geq 0.$$

Proof. Assume $W(\mathcal{S}, t) = W(\mathcal{P}, t) + KJ$. Then there are at least J jobs in system $\mathcal{S}$, so that all its servers are busy. Consequently, the coupling guarantees that if system $\mathcal{P}$ completes a task, so does system $\mathcal{S}$. Hence, in our coupling, it is not possible that $W(\mathcal{S}, t) > W(\mathcal{P}, t) + KJ$. ■

A direct consequence is as follows.

Lemma 2. *Let $c = 2 - 1/K$. It is possible to couple the systems so that*

$$X(\mathcal{S}, t) \leq X(\mathcal{P}, t) + cJ, \forall t \geq 0.$$

Proof. Assume $X(\mathcal{S}, t) > X(\mathcal{P}, t) + cJ$. Then,

$$\begin{aligned} W(\mathcal{S}, t) - W(\mathcal{P}, t) &= K(X(\mathcal{S}, t) - X(\mathcal{P}, t)) \\ &- \sum_{k=1}^{K-1} kY(\mathcal{S}, k, t) + \sum_{k=1}^{K-1} kY(\mathcal{P}, k, t) \\ &> KcJ - \sum_{k=1}^{K-1} kY(\mathcal{S}, k, t) \\ &\geq KcJ - (K-1)J \geq J[K(c-1)+1] \\ &= KJ, \end{aligned}$$

which contradicts the previous lemma.

In the derivation, we used two observations. First,

$$\begin{aligned} \sum_{k=1}^{K-1} kY(\mathcal{S}, k, t) &\leq (K-1) \sum_{k=1}^{K-1} Y(\mathcal{S}, k, t) \\ &\leq (K-1)J. \end{aligned}$$

This is the case because every job with a partial number of completed tasks occupies a server. Second,

$$K(c-1) + 1 = K$$

because $c = 2 - 1/K$.

■

We can now complete the proof of the theorem.

From Little's result, we have

$$\begin{aligned} D_S &= \frac{E(X(\mathcal{S}))}{\lambda} \\ &\leq \frac{E(X(\mathcal{P})) + cJ}{\lambda} \\ &= D_P + \frac{cJ}{\lambda}. \end{aligned}$$

■

2.6 Appendix

2.6.1 Proof of Proposition 1

Consider the queueing network of the system in the fluid limit. (See [16] for more discussion on the stability of fluid models) The fluid level of queue (k', k) at time t is

$$X_{(k',k)}(t) = \lim_{r\to\infty} \frac{Q_{(k',k)}(\lfloor rt \rfloor)}{r}.$$

The fluid limit dynamics is as follows. If k is a root node, then

$$X_{(0,k)}(t) = X_{(0,k)}(0) + A_{m(k)}(t) - D_k(t),$$

where $A_{m(k)}(t)$ is the total number of jobs of type m (scaled to the fluid level) that have arrived to the system until time t. If k is not a root node, then,

$$X_{(k',k)}(t) = X_{(k',k)}(0) + D_{k'}(t) - D_k(t),$$

where $D_k(t)$ is the total number of tasks (scaled to the fluid level) of type k processed up to time t. Suppose that $\rho^* > 1$. We show that if $X_{(k',k)}(0) = 0$ for all (k', k), there exists t_0 and (k', k) such that $X_{(k',k)}(t_0) \geq \epsilon(t_0) > 0$, which implies that the system is weakly unstable [17]. In contrary suppose that there exists a scheduling policy that under that policy for all $t \geq 0$ and all (k', k), $X_{(k',k)}(t) = 0$. Pick a regular point[6] t_1. Then, for all (k', k), $\dot{X}_{(k',k)}(t_1) = 0$. Since $\dot{A}_{m(k)}(t_1) = \lambda_m = \nu_k$, this implies that $\dot{D}_k(t_1) = \nu_k$ for all the root nodes k. Now considering queues (k', k) such that nodes k' are roots, one gets

$$\dot{D}_k(t_1) = \dot{D}_{k'}(t_1) = \nu_{k'} = \nu_k.$$

Similarly, one can inductively show that for all k, $\dot{D}_k(t_1) = \nu_k$. On the other hand, at a regular point t_1, $\dot{D}_k(t_1)$ is exactly the total service capacity allocated to task k at t_1. This implies that there exists p_{kj} at time t_1 such that $\nu_k = \sum_{j\in\mathcal{S}_k} \mu_{kj} p_{kj}$ for all k and the allocation vector $[p_{kj}]$ is feasible, i.e. $\sum_{k\in\mathcal{T}_j} p_{kj} \leq 1$. This contradicts $\rho^* > 1$.

Now suppose that $\rho^* \leq 1$, and $p^* = [p^*_{kj}]$ is an allocation vector that solves the LP. To prove sufficiency of the condition, consider a generalized head-of-the-line processor sharing policy that server j works on task k with capacity p^*_{kj}. Then the cumulative service allocated to task k up to time t is $S_k(t) = \sum_{j\in\mathcal{S}_k} \mu_{kj} p^*_{kj} t \geq \nu_k t$. We show that $X_{(k',k)}(t) = 0$ for all t and all (k', k), if $X_{(k',k)}(0) = 0$ for all (k', k). First consider queue $(0, k)$ corresponding to a root node. Suppose that $X_{(0,k)}(t_0) \geq \epsilon > 0$ for some positive t_0 and ϵ. Then, by continuity of fluid, there exists $0 < t_0 < t_1$ such that $\dot{X}_{(0,k)} > 0$. However, $\dot{X}_{(0,k)} = \nu_k - \sum_{j\in\mathcal{S}_k} \mu_{kj} p^*_{kj} \leq 0$, which is a contradiction. Now we show that $X_{(k',k)}(t) = 0$ for all t if k' is a root node and k is a child of k'. Note that $X_{(0,k')}(t) = 0$; thus, $\dot{D}_{k'}(t) = \nu_{k'} = \nu_k$. Then, $\dot{X}_{(k',k)} = \nu_k - \sum_{j\in\mathcal{S}_k} \mu_{kj} p^*_{kj} \leq 0$. This proves that $X_{(k',k)}(t) = 0$ for all t. One can then inductively complete this proof for all queues (k', k).

[6]We define a point t to be regular if $X_{(k',k)}(t)$ is differentiable at t for all (k', k).

2.6.2 Proof of Theorem 1

Recall the following notation which will be widely used in the proofs.

$$\mathbf{1}_{\{Q^n_{(k',k)}>0,\ \forall k'\in\mathcal{P}_k\}} = \mathbf{1}_{E^n_k}.$$

We introduce another notation which is

$$\mathbf{1}_{E^n} = \prod_{k=1}^{K} \mathbf{1}_{E^n_k}.$$

Note that event E^n denotes the event that all the queues are non-empty at time n.

Lemma 3. *Given any history $\mathcal{F}^n$ at time n, there exists a finite ℓ and $\delta_0 > 0$ such that $\mathbb{P}(\mathbf{1}_{E^{n+\ell}} = 1|\mathcal{F}^n) \geq \delta_0 > 0$.*

Proof. We work with each of the DAGs $\mathcal{G}_m$ separately, and construct events so that all the queues corresponding to $\mathcal{G}_m$ have positive lengths after some time ℓ. We can do this since $\mu_k p^n_k$ will always be no smaller than $\mu_k \varepsilon_0$ and strictly smaller than 1, so there is positive probability of serving or not serving a task.

Let $\widetilde{E}^n_k$ be the event that task k is served at time n, E^n_k be the event that task k is not served at time n, and $\hat{E}^n_m$ be the event that job type m arrives at time n. Consider a particular DAG $\mathcal{G}_m$. Recall that L_k is the length of the longest path from the root nodes of the DAG to node k. Let $\ell_m = \max_{k\in\mathcal{V}_m} L_k + 1$. We construct the event $E(\ell_m)$ that happens with a strictly positive probability, and assures that all the queues at time $n + \ell_m$ are non-empty. Toward this end, let $E(\ell_m) = \cap_{n'=0}^{\ell_m - 1} C^{n'}$, where event $C^{n'}$ is

$$C^{n'} = \hat{E}^{n'}_m \cap_{\{k:L_k\leq n'-1\}} \widetilde{E}^{n'}_k \cap_{\{k:L_k>n'-1\}} E^{n'}_k.$$

In words, $C^{n'}$ is the event that at time n', there is a job arrival of type m, services of tasks of class k for k with $L_k \leq n' - 1$, and no service of tasks of class k for k with $L_k > n' - 1$. Now, by construction all the queues are non-empty at time $n + \ell$ with a positive probability. To illustrate how we construct this event, consider the example of Figure 2.2 and the corresponding queueing network in Figure 2.3. Then, C^0 is the event that there is an arrival to the system, and no service in the network. C^1 is the event that there is a new job arriving, task 1 is served, and tasks 2, 3, and 4 are not served. Note that there is certainly at least one available task 1 to serve due to C^0. Up to now, certainly queues $(0, 1)$, $(1, 2)$, and $(1, 3)$ are non-empty. C^2 is the event of having a new arrival, service to tasks 1, 2, and 3, and no service to task 4. This construction assures that after 3 time slots, all the queues are non-empty.

Now for the whole network it is sufficient to take $\ell = \max_m \ell_m$. Construct the events $E(\ell_m)$ for each DAG independently, and freeze the DAG $\mathcal{G}_m$ (no service and no arrivals) from time $n + \ell_m$ to $n + \ell - 1$. This construction makes sure that all the queues in the network are non-empty at time $n + \ell$ given any history $\mathcal{F}^n$ with some positive probability δ_0. ■

Lemma 4. *The following inequality holds.*

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{n'=1}^{n} \mathbf{1}_{E^{n'}} \geq \frac{\delta_0}{\ell} > 0, a.s.$$

Proof. Take a subsequence $\mathbf{1}_{E^{n'\ell}}$, $n' \geq 1$. We couple the subsequence with an i.i.d. Bernoulli process $Y^{n'}$, $n' \geq 1$ with parameter δ_0 as follows. If $\mathbf{1}_{E^{n'\ell}} = 0$, then $Y^{n'} = 0$. If $\mathbf{1}_{E^{n'\ell}} = 1$, then $Y^{n'} = 1$ with probability

$$q_{n'} = \frac{\delta_0}{\mathbb{P}(\mathbf{1}_{E^{n'\ell}} = 1|\mathcal{F}^{(n'-1)\ell})},$$

and $Y^{n'} = 0$ with probability $1 - q_{n'}$. Note that by Lemma 3, $q_{n'} \leq 1$. $Y^{n'}$ is still marginally i.i.d. Bernoulli process with parameter δ_0. Then, with probability 1,

$$\begin{aligned}\liminf_{n\to\infty} \frac{1}{n}\sum_{n'=1}^{n} \mathbf{1}_{E^{n'}} &\geq \liminf_{n\to\infty} \frac{1}{n}\sum_{n'=1}^{\lfloor n/\ell\rfloor} \mathbf{1}_{E^{n'\ell}} \\ &\geq \liminf_{n\to\infty} \frac{1}{n}\sum_{n'=1}^{\lfloor n/\ell\rfloor} Y^{n'} = \frac{\delta_0}{\ell}.\end{aligned}$$

This completes the proof of Lemma 4. ■

Lemma 5. *The following equality holds.*

$$\lim_{n\to\infty} \sum_{n'=1}^{n} \beta^{n'}\mathbf{1}_{E^{n'}} = \infty, \ a.s.$$

Proof. From now on we work with the probability-1 event defined in Lemma 4. Consider a sample path in this probability-1 event, and let $x_{n'} = \beta^{n'}\mathbf{1}_{E^{n'}}$. First note that $x_{n'} \geq 0$. Thus, by the monotone convergence theorem, the series either converges or goes to infinity. Suppose that

$$\lim_{n\to\infty} \sum_{n'=1}^{n} x_{n'} = s$$

for some finite s. Define the sequence $b_{n'} = \frac{1}{\beta^{n'}}$. Then, by Kronecker's lemma [25], we have

$$\lim_{n\to\infty} \frac{1}{b_n}\sum_{n'=1}^{n} b_{n'}x_{n'} = 0.$$

This shows that

$$\lim_{n\to\infty} \frac{1}{b_n}\sum_{n'=1}^{n} \mathbf{1}_{E^{n'}} = 0,$$

which results in a contradiction, since $\lim_{n\to\infty} \frac{1}{n\beta^n}$ is finite, and hence by Lemma 4,

$$\liminf_{n\to\infty} \frac{1}{b_n} \sum_{n'=1}^{n} \mathbf{1}_{E^{n'}} > 0.$$

■

Now we are ready to prove Theorem 1. Consider the probability-1 event in Lemma 5. Let $d_n = \|p^n - p^*\|^2$ and fix $\epsilon > 0$. We prove that there exists a $n_0(\epsilon)$ such that for all $n \geq n_0(\epsilon)$, d_n has the following properties.

(i) If $d_n < \epsilon$, then $d_{n+1} < 3\epsilon$.

(ii) If $d_n \geq \epsilon$ then, $d_{n+1} \leq d_n - \gamma^n$ where $\sum_{n=1}^{\infty} \gamma^n = \infty$ and $\gamma^n \to 0$.

Then property (ii) shows that for some large enough $n_1 = n_1(\epsilon) > n_0(\epsilon)$, $d_{n_1} < \epsilon$, and properties (i) and (ii) show that $d_n < 3\epsilon$ for $n \geq n_1(\epsilon)$. This is true for all $\epsilon > 0$, so d_n converges to 0 almost surely.

First we show property (i). Let $U^n = [U_k^n] \in \mathbb{R}^K$ be the vector of updates such that

$$U_k^{n+1} = \mathbf{1}_{E_k^n} \sum_{(i',i)\in\mathcal{H}_k} \Delta Q_{(i',i)}^{n+1}.$$

Note that $\|U^n\|^2$ is bounded by some constant $C_1 > 0$, since the queues length changes at each time slot are bounded by 1. On the other hand, $\beta^n \to 0$. Thus, one can take $n_1(\epsilon)$ large enough such that for all $n \geq n_1(\epsilon)$, $\beta^n \leq \sqrt{\frac{\epsilon}{2C_1}}$. Then, for $n \geq n_1(\epsilon)$ if $d_n < \epsilon$,

$$\begin{aligned} d_{n+1} &= \|p^{n+1} - p^*\|^2 && (2.39)\\ &= \|[p^n + \beta^n U^{n+1}]_{\mathcal{C}_\varepsilon} - p^*\|^2 && (2.40)\\ &\leq \|p^n + \beta^n U^{n+1} - p^*\|^2 && (2.41)\\ &\leq 2d_n + 2(\beta^n)^2 \|U^{n+1}\|^2 < 3\epsilon, && (2.42) \end{aligned}$$

where (2.41) is due to the fact that projection to the convex set is non-expansive, and (2.42) is by Cauchy-Schwarz inequality.

To show property (ii), we make essential use of the fact that the cumulative stochastic noise is a martingale. Let

$$Z_k^{n+1} = \mathbf{1}_{E_k^n} (U_k^n - \nu_k + \mu_k p_k^n).$$

Then, by (2.10),

$$\mathbb{E}\left[Z_k^{n+1} | \mathcal{F}^n\right] = 0, \ \forall k \quad (2.43)$$

which shows that Z^n is a martingale difference sequence. Now observe that

$$\begin{aligned}
d_{n+1} &= \sum_{k=1}^{K}(p_k^{n+1} - p_k^*)^2 && (2.44)\\
&\le \sum_{k=1}^{K}(p_k^n + \beta^n U_k^{n+1} - p_k^*)^2 && (2.45)\\
&= d_n + (\beta^n)^2\|U^{n+1}\|^2 && (2.46)\\
&\quad + 2\beta^n \sum_{k=1}^{K}(p_k^n - p_k^*)(\nu_k - \mu_k p_k^n + Z_k^{n+1})\mathbf{1}_{E_k^n}\\
&= d_n + (\beta^n)^2\|U^{n+1}\|^2 && (2.47)\\
&\quad + 2\beta^n \sum_{k=1}^{K}(p_k^n - p_k^*)(\mu_k(p_k^* - p_k^n) + Z_k^{n+1})\mathbf{1}_{E_k^n}\\
&\le d_n + (\beta^n)^2 C_1 - \sum_{k=1}^{K} 2\mu_k\beta^n \mathbf{1}_{E^n}(p_k^n - p_k^*)^2 && (2.48)\\
&\quad + \sum_{k=1}^{K} 2\beta^n Z_k^{n+1}(p_k^n - p_k^*)\mathbf{1}_{E_k^n},
\end{aligned}$$

where (2.45) is due to non-expansiveness of projection, and (2.48) is due the facts that $E^n \subset E_k^n$ and $\|U^n\|^2 \le C_1$. Let $\mu^* = \min_k \mu_k$. Since $\sum_{k=1}^{K}(p_k^n - p_k^*)^2 > \epsilon$, the following choice of γ^n satisfies $d_{n+1} \le d_n - \gamma^n$:

$$\begin{aligned}
\gamma^n = &-(\beta^n)^2 C_1 + \beta^n \mathbf{1}_{E^n} 2\mu^*\epsilon\\
&- \sum_{k=1}^{K} 2\beta^n Z_k^{n+1}(p_k^n - p_k^*)\mathbf{1}_{E_k^n}.
\end{aligned}$$

As $\beta^n \to 0$ as $n \to \infty$, it is easy to see that $\gamma^n \to 0$ almost surely. Thus, to complete the proof of Theorem 1, one needs to show that $\sum_{n=1}^{\infty}\gamma_n = \infty$ almost surely. Toward this end, note that $\sum_n(\beta^n)^2$ is finite which makes $-\sum_n(\beta^n)^2 C_1$ bounded. By (2.43), and the facts that $\sum_n(\beta^n)^2 < \infty$ and $\|p^n - p^*\|$ is bounded for all n, we get that

$$V^n = \sum_{n'=1}^{n}\sum_{k=1}^{K} 2\beta^{n'} Z_k^{n'+1}(p_k^{n'} - p_k^*)\mathbf{1}_{E_k^{n'}}$$

is an L_2-bounded martingale and by the martingale convergence theorem converges to some bounded random variable almost surely [25]. Finally,

$$2\epsilon\mu^* \sum_{n=1}^{\infty}\beta^n \mathbf{1}_{E^n} = \infty, a.s.$$

by Lemma 5. This completes the proof of Theorem 1.

2.6.3 Proof of Theorem 2

In this subsection, we provide the proof of Theorem 2. The key idea to prove the rate stability of queues is to first show that the servers allocate enough cumulative capacity to all the tasks in the network. This is formalized in Lemma 6. Second, in Lemma 7, we show that each queue (k', k) cannot go unstable if task k receives enough service allocation over time, and the traffic rate coming to these queues is nominal. Finally, we use these two conditions to show rate stability of all the queues in the network by mathematical induction.

To prove the theorem, we first introduce some notation. Let D_k^n denote the cumulative number of processed tasks of type k at time n. Recall that d_k^n is the number of processed tasks of type k at time n. Therefore, $D_k^n = \sum_{n'=1}^n d_k^{n'}$. Let $A_m^n = \sum_{n'=1}^n a_m^{n'}$, $1 \leq m \leq M$ be the cumulative number of jobs of type m that have arrived up to time n. Then the queue-length dynamic of queue (k', k) can be written as follows. If $k' \neq 0$, then $Q_{(k',k)}^n = Q_{(k',k)}^0 + D_{k'}^n - D_k^n$. If $k' = 0$, then $Q_{(k',k)}^n = Q_{(k',k)}^0 + A_{m(k)}^n - D_k^n$.

At time n, the probability that one task is served and departed from queue (k', k) is $\mu_k p_k^n$, if all of the queues (i, k) are non-empty for all i. We define s_k^n to be a random variable denoting the virtual service that queues (k', k) have received at time n, whether there has been an available task k to be processed or not. s_k^n is a Bernoulli random variable with parameter $\mu_k p_k^n$. Then, the cumulative service that queues (k', k) receive up to time n is $S_k^n = \sum_{n'=1}^n s_k^{n'}$ for all k'. Note that the cumulative service is different from the cumulative departure. Indeed, $d_k^n = s_k^n \mathbf{1}_{E_k^n}$.

From now on, in the proof of Theorem 2, we consider the probability-1 event that p^n converges to p^* stated in Theorem 1.

Lemma 6. *The following equality holds:*

$$\lim_{n\to\infty} \frac{S_k^n}{n} = \nu_k, \ a.s., \ \forall k. \tag{2.49}$$

Proof. By Theorem 1, the sequence p_k^n converges to $\frac{\nu_k}{\mu_k}$ almost surely. Therefore, for all the sample paths in the probability-1 event, and for all $\epsilon_1 > 0$, there exists $n_0(\epsilon_1)$ such that $\|\mu_k p_k^n - \nu_k\| \leq \epsilon_1$, for all $n > n_0(\epsilon_1)$.

Let $\tilde{s}_k^n$ be i.i.d Bernoulli process of parameter $\nu_k - \epsilon_1$. We couple the processes s_k^n and $\tilde{s}_k^n$ as follows. If $s_k^n = 0$, then $\tilde{s}_k^n = 0$. If $s_k^n = 1$, then $\tilde{s}_k^n = 1$ with probability $\frac{\nu_k - \epsilon_1}{\mu_k p_k^n}$, and $\tilde{s}_k^n = 0$ with probability $1 - \frac{\nu_k - \epsilon_1}{\mu_k p_k^n}$. Note that $\tilde{s}_k^n$ is still marginally i.i.d Bernoulli process of parameter $\nu_k - \epsilon_1$. Then,

$$\liminf_{n\to\infty} \frac{S_k^n}{n} \geq \liminf_{n\to\infty} \frac{\sum_{n'=n_0(\epsilon_1)+1}^n s_k^{n'}}{n} \tag{2.50}$$

$$\geq \liminf_{n\to\infty} \frac{\sum_{n'=n_0(\epsilon_1)+1}^n \tilde{s}_k^{n'}}{n} \tag{2.51}$$

$$= \nu_k - \epsilon_1 \ a.s., \tag{2.52}$$

where (2.51) is by construction of the coupled sequences, and (2.52) is by the strong law of large numbers.

Let $\bar{s}_k^n$ be i.i.d Bernoulli process of parameter $\nu_k + \epsilon_1$. We couple the processes s_k^n and $\bar{s}_k^n$ as follows. If $s_k^n = 1$, then $\tilde{s}_k^n = 1$. If $s_k^n = 0$, then $\tilde{s}_k^n = 0$ with probability $\frac{1-(\nu_k+\epsilon_1)}{1-\mu_k p_k^n}$, and $\tilde{s}_k^n = 1$ with probability $1 - \frac{1-(\nu_k+\epsilon_1)}{1-\mu_k p_k^n}$. Note that $\bar{s}_k^n$ is still marginally i.i.d Bernoulli process of parameter $\nu_k + \epsilon_1$. Then,

$$\limsup_{n\to\infty} \frac{S_k^n}{n} \le \limsup_{n\to\infty} \frac{n_0(\epsilon_1) + \sum_{n'=n_0(\epsilon_1)+1}^{n} s_k^{n'}}{n} \tag{2.53}$$

$$\le \limsup_{n\to\infty} \frac{n_0(\epsilon_1) + \sum_{n'=n_0(\epsilon_1)+1}^{n} \bar{s}_k^{n'}}{n} \tag{2.54}$$

$$= \nu_k + \epsilon_1 \ a.s., \tag{2.55}$$

where (2.54) is by construction of the coupled sequences, and (2.55) is by the strong law of large numbers. Letting $\epsilon_1 \to 0$, we have

$$\liminf_{n\to\infty} \frac{S_k^n}{n} = \limsup_{n\to\infty} \frac{S_k^n}{n} = \nu_k, \ a.s.$$

■

Lemma 7. *Consider a fixed k and all queues (k', k) with $k' \in \mathcal{P}_k$. Suppose that*

$$\lim_{n\to\infty} \frac{D_{k'}^n}{n} = \nu_k, \ \forall k' \in \mathcal{P}_k, \ a.s. \tag{2.56}$$

if $\mathcal{P}_k \neq \emptyset$, and

$$\lim_{n\to\infty} \frac{A_{m(k)}^n}{n} = \nu_k, \ a.s. \tag{2.57}$$

if k is a root node. Then,

$$\lim_{n\to\infty} \frac{Q_{(k',k)}^n}{n} = 0, a.s.$$

Proof. Before getting to the details of the proof, note that if k is a root node, then we readily know that

$$\lim_{n\to\infty} \frac{A_{m(k)}^n}{n} = \lambda_m = \nu_k, a.s.$$

Thus, the lemma states that queues $(0, k)$ are rate stable.

We prove the lemma for the general case that node k is not a root node. Similar proof holds for the case of root nodes. First, we show that for all pair of queues (i, k) and (i', k) such that $i, i' \in \mathcal{P}_k$, we have

$$\lim_{n\to\infty} \frac{Q_{(i,k)}^n - Q_{(i',k)}^n}{n} = 0, a.s. \tag{2.58}$$

Note that

$$\begin{aligned}\frac{Q^n_{(i,k)} - Q^n_{(i',k)}}{n} &= \frac{D^n_i - D^n_k - (D^n_{i'} - D^n_k)}{n} \\ &= \frac{D^n_{i'} - D^n_i}{n}.\end{aligned}$$

Then, by (2.56),

$$\lim_{n\to\infty} \frac{D^n_i - D^n_{i'}}{n} = \nu_k - \nu_k = 0, a.s.$$

Second, we show that

$$\liminf_{n\to\infty} \frac{Q^n_{(k',k)}}{n} = 0, a.s.$$

In contrary suppose that in one realization in the probability-1 event defined by (2.56) and Lemma 6,

$$\liminf_{n\to\infty} \frac{Q^n_{(k',k)}}{n} > 2\epsilon_2, \tag{2.59}$$

for some $\epsilon_2 > 0$. This implies that in that realization,

$$\liminf_{n\to\infty} \frac{Q^0_{(k',k)} + D^n_{k'} - D^n_k}{n} > 2\epsilon_2.$$

By (2.56), the probability that $\lim_{n\to\infty} \frac{Q^0_{(k',k)}+D^n_{k'}}{n} = \nu_k$ is 1. Thus, in that realization

$$\limsup_{n\to\infty} \frac{D^n_k}{n} < \nu_k - 2\epsilon_2. \tag{2.60}$$

On the other hand, (2.59) shows that there exists $n_0(\epsilon_2)$ such that for all $n \geq n_0(\epsilon_2)$,

$$Q^n_{(k',k)} \geq 2n\epsilon_2. \tag{2.61}$$

Furthermore, (2.58) shows that there exists $n_1(\epsilon_2)$ such that for all $i \in \mathcal{P}_k$ and for all $n \geq n_1(\epsilon)$,

$$|Q^n_{(k',k)} - Q^n_{(i,k)}| < n\epsilon_2. \tag{2.62}$$

Let $n_2(\epsilon_2) = \max(n_0(\epsilon_2), n_1(\epsilon_2))$. (2.61) and (2.62) imply that for all $n \geq n_2(\epsilon_2)$, $Q^n_{(i,k)} \geq n\epsilon_2$. Now taking $n_3(\epsilon_2) = \max(n_2(\epsilon_2), 1/\epsilon_2)$, we have that all the queues (i,k), $i \in \mathcal{P}_k$ are non-empty for $n \geq n_3(\epsilon_2)$. Thus, $s^n_k = d^n_k$ for all $n \geq n_3(\epsilon_2)$. Therefore,

$$\begin{aligned}\limsup_{n\to\infty} \frac{S^n_k}{n} &\leq \limsup_{n\to\infty} \frac{n_3(\epsilon_2) + D^n_k}{n} \\ &= \limsup_{n\to\infty} \frac{D^n_k}{n}.\end{aligned}$$

Thus, by Lemma 6, $\nu_k \leq \limsup_{n\to\infty} \frac{D_k^n}{n}$ which contradicts (2.60). Since this holds for any $\epsilon_2 > 0$, we conclude that

$$\liminf_{n\to\infty} \frac{Q^n_{(k',k)}}{n} = 0, \;\; a.s., \tag{2.63}$$

for all $k' \in \mathcal{P}_k$.

Third, we show that

$$\limsup_{n\to\infty} \frac{Q^n_{(k',k)}}{n} = 0, a.s.$$

In contrary suppose that in one realization in the probability-1 event defined by (2.56), (2.58), and Lemma 6,

$$\limsup_{n\to\infty} \frac{Q^n_{(k',k)}}{n} > 4\epsilon_3, \tag{2.64}$$

for some $\epsilon_3 > 0$. This implies that in that realization $Q^n_{(k',k)} > 4n\epsilon_3$ happens infinitely often. Moreover, by (2.63), $Q^n_{(k',k)} < 2n\epsilon_3$ happens also infinitely often in that realization. On the other hand, by (2.58), there exists some $n_0(\epsilon_3)$ such that for all $n \geq n_0(\epsilon_3)$ and all $i \in \mathcal{P}_k$,

$$|Q^n_{(i,k)} - Q^n_{(k',k)}| < n\epsilon_3. \tag{2.65}$$

Take $N_1 = \max(n_0(\epsilon_3), \frac{2}{\epsilon_3})$. Then, there exists $N_1 \leq n_1(\epsilon_3) < n_2(\epsilon_3)$ such that $Q^{n_1}_{(k',k)} \leq 2n_1\epsilon_3$ and $Q^{n_2}_{(k',k)} \geq 4n_2\epsilon_3$. In words, n_1+1 is the first time after N_1 that $\frac{Q^n_{(k',k)}}{n}$ crosses $2\epsilon_3$ without going below $2\epsilon_3$ before exceeding $4\epsilon_3$. Then, since the queue-length changes by at most 1 each time slot, queue (k',k) is non-empty for all n, $n_1 \leq n \leq n_2$. Furthermore, for all n, $n_1 \leq n \leq n_2$ and for all $i \in \mathcal{P}_k$, by (2.65),

$$\begin{aligned} Q^n_{(i,k)} &> Q^n_{(k',k)} - n\epsilon_3 \\ &\geq 2n\epsilon_3 - 1 - n\epsilon_3 \\ &\geq n_1\epsilon_3 - 1 \geq 1. \end{aligned}$$

Thus, all the queues (i,k) are also non-empty for all n in the interval $n_1 \leq n \leq n_2$. Consequently, for all n, $n_1 \leq n \leq n_2$, $s_k^n = d_k^n$. Now define a process

$$B^n_{(k',k)} = D^n_{k'} - S^n_k.$$

Note that by (2.56) and Lemma 6, in the realization of probability-1 event that we consider,

$$\lim_{n\to\infty} \frac{B^n_{(k',k)}}{n} = \nu_k - \nu_k = 0. \tag{2.66}$$

We bound $B^{n_2}_{(k',k)}$ as follows.

$$\begin{aligned}
B^{n_2}_{(k',k)} &= B^{n_1}_{(k',k)} + [B^{n_2}_{(k',k)} - B^{n_1}_{(k',k)}] \\
&= B^{n_1}_{(k',k)} + [D^{n_2}_{k'} - D^{n_1}_{k'} - (S^{n_2}_{k} - S^{n_1}_{k})] \\
&= B^{n_1}_{(k',k)} + [D^{n_2}_{k'} - D^{n_1}_{k'} - (D^{n_2}_{k} - D^{n_1}_{k})] \\
&= B^{n_1}_{(k',k)} + [Q^{n_2}_{k} - Q^{n_1}_{k}] \\
&\geq B^{n_1}_{(k',k)} + 4\epsilon_3 n_2 - 2\epsilon_3 n_1.
\end{aligned}$$

Dividing both sides of the inequality by n_2 and subtracting $B^{n_1}_{(k',k)}/n_1$, one gets

$$\frac{B^{n_2}_{(k',k)}}{n_2} - \frac{B^{n_1}_{(k',k)}}{n_1} \geq \frac{B^{n_1}_{(k',k)}}{n_1}(\frac{n_1}{n_2} - 1) + 4\epsilon_3 - 2\epsilon_3\frac{n_1}{n_2}.$$

By (2.66), one can choose a large enough N_2 such that for all $n \geq N_2$, $|\frac{B^{n}_{(k',k)}}{n}| \leq \frac{2\epsilon_3}{3}$. Then, N_1 can be chosen as

$$N_1 = \max(n_0(\epsilon_3), \frac{2}{\epsilon_3}, N_2),$$

and one chooses n_1 and n_2 accordingly as before. Then, since $n_1, n_2 \geq N_1$, one can write

$$|\frac{B^{n}_{(k',k)}}{n} - \frac{B^{n}_{(k',k)}}{n}| \leq \frac{4\epsilon_3}{3}. \tag{2.67}$$

However,

$$\frac{B^{n_2}_{(k',k)}}{n_2} - \frac{B^{n_1}_{(k',k)}}{n_1} \geq 2\epsilon_3(\frac{n_1}{n_2} - 1) + 4\epsilon_3 - 2\epsilon_3\frac{n_1}{n_2} = 2\epsilon_3,$$

which contradicts (2.67). Thus,

$$\limsup_{n\to\infty} \frac{Q^{n}_{(k',k)}}{n} = 0, a.s.$$

The result holds for arbitrary $k' \in \mathcal{P}_k$. This completes the proof of Lemma 7. ■

Now we are ready to prove Theorem 2. We complete the proof of Theorem 2 by induction. Recall that L_k is the length of the longest path from the root of the DAG $\mathcal{G}_{m(k)}$ to node k. If k is a root, $L_k = 0$. The formal induction goes as follows.

- **Basis:** All the queues corresponding to root nodes, i.e. all (k', k) for which $L_k = 0$ are rate stable.
- **Inductive Step:** If all the queues (k', k) for which $L_k \leq L - 1$ are rate stable, then all the queues (k', k) for which $L_k = L$ are also rate stable.

The basis is true by Lemma 7. The inductive step is also easy to show using Lemma 7. For a particular queue $(k', k) = (i_L, i_{L+1})$, suppose that $L_k = L$. Pick a path of edges

$$(i_1, i_2), (i_2, i_3), \ldots, (i_L, i_{L+1}),$$

from queue $(0, i_1)$ to (k', k). By assumption of induction, all the queues (i_l, i_{l+1}) are rate stable for $l \leq L - 1$.

$$\begin{aligned} A^n_{m(k)} - D^n_{i_L} &= A^n_{m(k)} - D^n_{i_1} + \sum_{l=1}^{L-1}(D^n_{i_l} - D^n_{i_{l+1}}) \\ &= Q^n_{(0,i_1)} - Q^0_{(0,i_1)} + \sum_{l=1}^{L-1}(Q^n_{(i_l,i_{l+1})} - Q^0_{(i_l,i_{l+1})}). \end{aligned}$$

Therefore,

$$\begin{aligned} \lim_{n\to\infty} \frac{D^n_{i_L}}{n} = \lim_{n\to\infty} \Big[& \frac{A^n_{m(k)}}{n} - \frac{Q^n_{(0,i_1)} - Q^0_{(0,i_1)}}{n} \\ & - \sum_{l=1}^{L-1} \frac{(Q^n_{(i_l,i_{l+1})} - Q^0_{(i_l,i_{l+1})})}{n} \Big] \\ = {} & \lambda_m = \nu_k, a.s. \end{aligned}$$

Now since $\lim_{n\to\infty} \frac{D^n_{k'}}{n} = \nu_k$ *a.s.*, by Lemma 7, (k', k) is rate stable. This completes the proof of the induction step and as a result the proof of Theorem 2.

2.6.4 Proof of Theorem 4

Let D^n_k denote the cumulative number of tasks that have departed queue k by and including time n: $D^n_k = \sum_{n'=1}^{n} d^{n'}_k$. Define s^n_k to be a random variable denoting the virtual service that queue k receives at time n, whether the queue has been empty or not. s^n_k is a Bernoulli random variable with parameter $\mu_k p^n_k$. Note that $d^n_k = s^n_k \mathbf{1}_{Q^n_k > 0}$. Define the cumulative service that queue k has received up to time n to be $S^n_k = \sum_{n'=1}^{n} s^n_k$.

Lemma 8. *The following equality holds:*

$$\lim_{n\to\infty} \frac{\sum_{n'=1}^{n} s^{n'}_k \mathbf{1}^{n'}_{k\to k'}}{n} = \nu_k r_{kk'}, \; a.s., \; \forall k, k'. \tag{2.68}$$

Proof. First note that the sequence of random variables $\mathbf{1}^{n'}_{k\to k'}$ is i.i.d. Bernoulli-distributed with parameter $r_{kk'}$, and independent of the sequence $s^{n'}_k$. Now by Theorem 3, the sequence $\mu_k p^{n'}_k$ converges to ν_k almost surely. Thus, in this probability-1 event, for all $\epsilon_4 > 0$, there exists $n_0(\epsilon_4)$ such that $\|\mu_k p^{n'}_k - \nu_k\| \leq \epsilon_4$ for all $n' > n_0(\epsilon_4)$.

Let w_k^n be i.i.d Bernoulli process of parameter $(\nu_k - \epsilon_4)r_{kk'}$. We couple the processes $s_k^n \mathbf{1}_{k\to k'}^n$ and w_k^n as follows. If $s_k^n = 0$, then $w_k^n = 0$. If $s_k^n = 1$, then $w_k^n = \mathbf{1}_{k\to k'}^n$ with probability $\frac{\nu_k - \epsilon_4}{\mu_k p_k^n}$, and $w_k^n = 0$ with probability $1 - \frac{\nu_k-\epsilon_4}{\mu_k p_k^n}$. w_k^n is still marginally i.i.d. Bernoulli process of parameter $(\nu_k - \epsilon_4)r_{kk'}$. Then,

$$\begin{aligned}\liminf_{n\to\infty} \frac{\sum_{n'=1}^n s_k^{n'} \mathbf{1}_{k\to k'}^{n'}}{n} &\geq \liminf_{n\to\infty} \frac{\sum_{n'=n_0(\epsilon_4)+1}^n w_k^{n'}}{n} \\ &= (\nu_k - \epsilon_4)r_{kk'} \ a.s.\end{aligned}$$

Now we couple the processes $s_k^n \mathbf{1}_{k\to k'}^n$ and v_k^n, where v_k^n is i.i.d Bernoulli process of parameter $(\nu_k - \epsilon_4)r_{kk'}$. If $s_k^n = 1$, then $v_k^n = \mathbf{1}_{k\to k'}^n$. If $s_k^n = 0$, then $v_k^n = \mathbf{1}_{k\to k'}^n$ with probability $\frac{1-(\nu_k+\epsilon_4)}{1-\mu_k p_k^n}$, and $v_k^n = 0$ with probability $1 - \frac{1-(\nu_k+\epsilon_4)}{1-\mu_k p_k^n} r_{kk'}$. v_k^n is still marginally i.i.d. Bernoulli process of parameter $(\nu_k + \epsilon_4)r_{kk'}$. Then,

$$\begin{aligned}\limsup_{n\to\infty} \frac{\sum_{n'=1}^n s_k^{n'} \mathbf{1}_{k\to k'}^{n'}}{n} &\leq \limsup_{n\to\infty} \frac{\sum_{n'=n_0(\epsilon_4)+1}^n v_k^{n'}}{n} \\ &= (\nu_k + \epsilon_4)r_{kk'} \ a.s.\end{aligned}$$

The proof is complete by letting $\epsilon_4 \to 0$. ■

Now we are ready to complete the proof of the theorem. Observe that

$$\begin{aligned}Q_k^n &= Q_k^0 + A_k^n + \sum_{n'=1}^n \sum_{k'=1}^K d_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n \\ &\leq Q_k^0 + A_k^n + \sum_{n'=1}^n \sum_{k'=1}^K s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n.\end{aligned}$$

So it is enough to show that

$$\lim_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^n \sum_{k'=1}^K s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} = 0.$$

First, we show that

$$\liminf_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^n \sum_{k'=1}^K s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} = 0, \ a.s.$$

In contrary suppose that in a realization,

$$\liminf_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^n \sum_{k'=1}^K s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} > \epsilon_5,$$

for some $\epsilon_5 > 0$. Then, using Lemma 8 and the fact that $\lim_{n\to\infty} \frac{A_k^n}{n} = \lambda_k$, we have

$$\limsup_{n\to\infty} \frac{D_k^n}{n} < \lambda_k + \sum_{k'=1}^{K} \nu_{k'} r_{k'k} - \epsilon_5 = \nu_k - \epsilon_5. \tag{2.69}$$

On the other hand, $\liminf_{n\to\infty} \frac{Q_k^n}{n} > \epsilon_5$ implies that there exists $n_0(\epsilon_5) > \frac{1}{\epsilon_5}$ such that for all $n > n_0(\epsilon_5)$, $Q_k^n \geq \epsilon_5 n$, or in words, the queue is non-empty after $n_0(\epsilon_5)$. Thus, $s_k^{n'} = d_k^{n'}$ for $n' > n_0$. Therefore,

$$\begin{aligned}\limsup_{n\to\infty} \frac{S_k^n}{n} &\leq \limsup_{n\to\infty} \frac{n_0 + D_k^n}{n} \\ &= \limsup_{n\to\infty} \frac{D_k^n}{n}.\end{aligned}$$

Now by Lemma 6, $\limsup_{n\to\infty} \frac{S_k^n}{n} = \nu_k \leq \limsup_{n\to\infty} \frac{D_k^n}{n}$ which contradicts (2.69).[7] Since this holds for any $\epsilon_5 > 0$, we conclude that

$$\liminf_{n\to\infty} \frac{Q_k^n}{n} = 0, \ \ a.s. \tag{2.70}$$

Second, we show that

$$\limsup_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^{n} \sum_{k'=1}^{K} s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} = 0, a.s.$$

Suppose that in a realization

$$\limsup_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^{n} \sum_{k'=1}^{K} s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} > 2\epsilon_6,$$

for some $\epsilon_6 > 0$. This implies that in this realization, $Q_k^n > 2\epsilon_6 n$ happens infinitely often and

$$\limsup_{n\to\infty} \frac{A_k^n + \sum_{n'=1}^{n} \sum_{k'=1}^{K} s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} > 2\epsilon_6$$

in that realization. Moreover, by (2.70), for any $\epsilon_6 > 0$, $Q_k^n < \epsilon_6 n$ happens infinitely often with probability 1. Let $N_2 \geq \frac{2}{\epsilon_6}$. Then, there exist $N_2 \leq n_3 < n_4$ such that $Q_k^{n_3} \leq \epsilon_6 n_3$ and $Q_k^{n_4} \geq 2\epsilon_6 n_4$ and queue k is non-empty between time n_3 and n_4. Define a process

$$\tilde{B}_k^n = A_k^n + \sum_{n'=1}^{n} \sum_{k'=1}^{K} s_{k'}^{n'} \mathbf{1}_{k'\to k}^{n'} - S_k^n.$$

[7]The lemma is also valid for the flexible queueing network, and the proof does not change.

Then,

$$\begin{aligned}\tilde{B}_k^{n_4} &= \tilde{B}_k^{n_3} + [\tilde{B}_k^{n_4} - \tilde{B}_k^{n_3}] &(2.71)\\ &\geq \tilde{B}_k^{n_3} + Q_k^{n_4} - Q_k^{n_3} &(2.72)\\ &\geq \tilde{B}_k^{n_3} + 2\epsilon_6 n_4 - \epsilon_6 n_3. &(2.73)\end{aligned}$$

(2.72) is due to the following.

$$\begin{aligned}&\tilde{B}_k^{n_4} - \tilde{B}_k^{n_3}\\ &= A_k^{n_4} - A_k^{n_3} + \sum_{n'=n_3+1}^{n_4}\sum_{k'=1}^{K} s_{k'}^{n'}\mathbf{1}_{k'\to k}^{n'} - (S_k^{n_4} - S_k^{n_3})\\ &\geq A_k^{n_4} - A_k^{n_3} + \sum_{n'=n_3+1}^{n_4}\sum_{k'=1}^{K} d_{k'}^{n'}\mathbf{1}_{k'\to k}^{n'} - (S_k^{n_4} - S_k^{n_3})\\ &= A_k^{n_4} - A_k^{n_3} + \sum_{n'=n_3+1}^{n_4}\sum_{k'=1}^{K} d_{k'}^{n'}\mathbf{1}_{k'\to k}^{n'} - (D_k^{n_4} - D_k^{n_3}) \qquad (2.74)\\ &= Q_k^{n_4} - Q_k^{n_3}.\end{aligned}$$

(2.74) is true since queue k is non-empty between time n_3 and n_4. Now (2.73) implies that

$$\frac{\tilde{B}_k^{n_4}}{n_4} - \frac{\tilde{B}_k^{n_3}}{n_3} \geq \frac{\tilde{B}_k^{n_3}}{n_3}(\frac{n_3}{n_4} - 1) + 2\epsilon_6 - \epsilon_6\frac{n_3}{n_4}.$$

By Lemma 6 we know that

$$\lim_{n\to\infty}\frac{\tilde{B}_k^n}{n} = 0 \ a.s.,$$

so one can choose N_2 large enough such that for all $n \geq N_2$, $|\tilde{B}_k^n/n| \leq \epsilon_6/3$. Then,

$$|\frac{\tilde{B}_k^{n_4}}{n_4} - \frac{\tilde{B}_k^{n_3}}{n_3}| \leq \frac{2\epsilon_6}{3}. \qquad (2.75)$$

However, since $\frac{\tilde{B}_k^{n_3}}{n_3} \leq \epsilon_6$ and $\frac{n_3}{n_4} < 1$,

$$\frac{\tilde{B}_k^{n_4}}{n_4} - \frac{\tilde{B}_k^{n_3}}{n_3} \geq \epsilon_6(\frac{n_3}{n_4} - 1) + 2\epsilon_6 - \epsilon_6\frac{n_3}{n_4} = \epsilon_6,$$

which contradicts (2.75). Thus,

$$\limsup_{n\to\infty}\frac{A_k^n + \sum_{n'=1}^{n}\sum_{k'=1}^{K} s_{k'}^{n'}\mathbf{1}_{k'\to k}^{n'} - D_k^n}{n} = 0, a.s.,$$

which completes the proof Theorem 4.

Chapter 3

Stability of Multiclass Queueing Networks under Longest-Queue Scheduling

3.1 Introduction

In Chapter 2, we considered the problem of how to design robust scheduling policies. Specifically, we focused on two different network models: fork-join network for processing DAGs and flexible queueing network. As mentioned before, it is of great practical interest to find scheduling policies that are simple, robust, and local. A natural low-complexity scheduling algorithm is longest-queue (LQ) scheduling, studied in [65], [50] and [24], which we discuss in this chapter in detail.

In this chapter, we consider a simpler network model: the open multiclass queueing network [34], [46], [17]. These are queueing networks with disjoint groups of queues that cannot be scheduled simultaneously, and therefore, they are special cases of flexible queueing networks studied in Chapter 2, Section 2.3. More specifically, each queue in this network can be only served by 1 server, so the jobs in the queues are *not* flexible.

In general, we know that the utilization being less than one for each server is necessary but not sufficient for the stability of a queueing network. This condition specifies that work arrives at each server at rate less than one. In [46], Lu and Kumar provided an example of a network (Figure 3.1) with priority scheduling that is unstable despite satisfying the utilization condition. The priority is given to queue 2 in group 1 and to queue 3 in group 2. To see this, assume that $\mu_1 > \mu_3$ and $\mu_4 > \mu_2$ and that queue 3 is initially empty while queue 2 is not. Group 1 serves queue 2, so that queue 1 is not served and queue 3 remains empty. Eventually, queue 2 becomes empty and queues 1 and 3 get served until queue 3 becomes empty (because $\mu_1 > \mu_3$). At that time, queues 4 and 2 get served, until queue 2 becomes empty (because $\mu_4 > \mu_2$). Thus, queues 2 and 3 are never served together. Consequently, they form a virtual

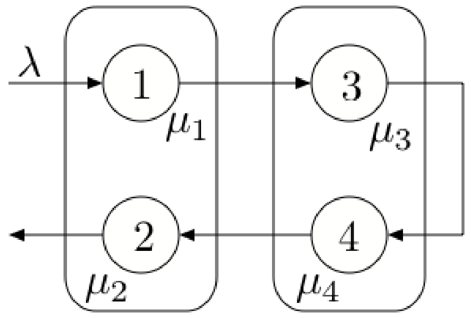

Figure 3.1: Lu-Kumar network

group and the system cannot be stable unless the utilization of that virtual group is less than one, which is an additional condition not implied by the original utilization condition. For more details about the proofs, see Lemma 4.1.2 in [17].

If priority is given to the last buffer in each group (queues 2 and 4) or first buffer in each group (queues 1 and 3), the network is stable. It is shown in [20, 45, 44] that, for a re-entrant line, the last-buffer-first-serve and first-buffer-first serve discipline with pre-emption is throughput-optimal. Max-Weight scheduling, proposed in [66, 18], is known to be throughput-optimal for flexible queueing networks; thus, it is also throughput-optimal for open multiclass queueing networks. However, Max-Weight scheduling suffers from dependency on the knowledge of all of the service rates and routing probabilities. Furthermore, scheduling decisions per server can not be done locally, and the knowledge of the queue-lengths in the network can be required.

In this chapter, we analyze the stability of the queueing networks under LQ scheduling policy using their fluid model. The fluid model converts a stochastic system into a deterministic system, based on the functional strong law of large numbers, [47, 16, 63, 17, 15]. Under weak assumptions, the stability of the fluid model implies the stability of the queueing network.

There has been relatively little work on the stability of LQ scheduling in the literature. We can mainly mention the following papers. In [24], Dimakis and Walrand identify new sufficient conditions for longest-queue-first (LQF) scheduling to be throughput-optimal. They combine properties of diffusion-scaled path functionals and local fluid limits into a sharper characterization of stability. See also [40] for a variation on the first order sufficient condition of that paper (resource pooling). In [9], Baharian and Tezcan consider LQF scheduling for parallel server systems. It is shown that the nominal traffic condition is sufficient to prove stability if the underlying graph of the parallel server system is a tree. Furthermore, additional drift conditions are provided for the stability of a special parallel server system known as X-model (see Figure 2.9). The network model that we consider is different from all the previous work on LQ scheduling. In fact, throughput-optimality of LQ scheduling for open multiclass queueing networks is still and open problem.

In addition to LQ scheduling, we propose a new scheduling policy called longest-dominating-queue (LDQ) in Section 3.4. This policy is closely related to LQ scheduling. According to this policy, none of the queues that feed a larger maximum-length queue in the network is served. Among those queues that are not dominated by a maximum-length queue in the network, the longest one is scheduled. We use the fluid model and show that the maximum of the queue lengths is a Lyapunov function to prove the stability of LDQ scheduling in a general network which is an acyclic graph.

The rest of this chapter is organized as follows. In Section 3.2, we provide the precise network model and problem formulation. In Section 3.3, we focus on the LQ scheduling and prove that it is throughput-optimal for multiclass queueing networks with two groups of two queues. In Section 3.4, we propose the LDQ scheduling and prove that this new policy is maximally stable if the network topology is acyclic. We provide simulation results and examples, which show that when there is a cycle in the network, LDQ may not be stable.

3.2 Problem Definition and Network Model

In this section, we first introduce open multiclass queueing networks. We discuss the fluid model to analyze the stability of the queueing network. We also provide the utilization condition or capacity region of the network.

3.2.1 Network Model

We consider a network with K queues and a routing matrix $R_{K\times K}$. The entry r_{ik} is the probability that a job goes to queue k upon leaving queue i . Therefore, a job leaves the network upon leaving queue i with probability $1-\sum_k r_{ik}$. In each queue, jobs are served in their order of arrival. A queueing network is "acyclic" if a job cannot visit any queue more than once; that is, if there exists no finite sequence of queues $(i_1, i_2, \dots, i_\ell, i_1)$ such that $r_{i_1i_2}r_{i_2i_3}\cdots r_{i_\ell i_1} > 0$.

The random length of queue i at time $t \geq 0$ is $\bar{X}_i(t)$, the vector of service rates of the queues is $\mu = [\mu_1, \mu_2, \dots, \mu_K]^T$, and the vector of arrival rates to the queues is $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_K]^T$. The arrival processes to the queues are independent stationary ergodic processes. In this network, not all the queues can be served at the same time. The queues are partitioned into J disjoint groups $\{\mathcal{G}_j\}_{j=1}^J$ and only one queue can be served in each group at a time. Let $K_j = |\mathcal{G}_j|$, so that $K_1 + K_2 + \cdots + K_J = K$. Figure 3.2 illustrates a multiclass queueing network with 2 groups of two queues: $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{3, 4\}$.

We assume the network is open, i.e., all the jobs eventually leave the network. Since the network is open,

$$Q = I + \sum_{k=1}^{\infty}(R^T)^k = (I - R^T)^{-1}$$

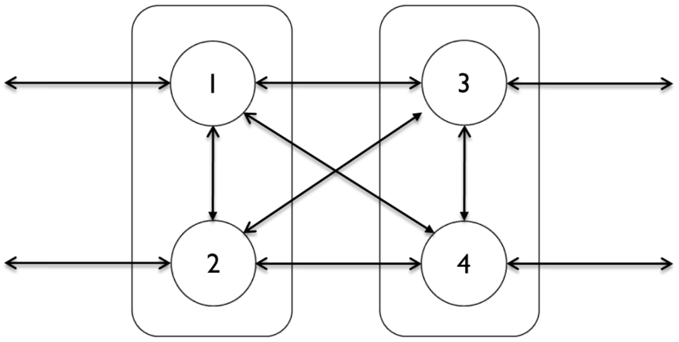

Figure 3.2: Multiclass queueing network with two groups of two queues

is a finite positive matrix. So, the matrix $(I - R^T)$ is invertible.

The service disciplines that we consider are independent of λ, μ and R. The goal is to analyze the stability of this network when the utilization conditions stated in Section 3.2.2 hold. Let $\bar{X}_i(t)$ be the length of queue i at time t and $\bar{T}_i(t)$ be the random total amount of time that queue i has been scheduled up to time t. We use the fluid model to analyze the stability of the network. The fluid model is described by a set of differential equations as follows. Let $X_i(t) = \lim_{r\to\infty} \frac{\bar{X}_i(rt)}{r}$ be the fluid level of queue i at time t, and let $T_i(t) = \lim_{r\to\infty} \frac{\bar{T}_i(rt)}{r}$ be the cumulative amount of time that queue i is served up to time t in the fluid model. Then the fluid model equations are

$$X_k(t) = X_k(0) + \lambda_k t - \mu_k T_k(t) + \sum_{i=1}^{K} r_{ik}\mu_i T_i(t), \quad k = 1, \ldots, K. \tag{3.1}$$

The fluid model is *stable* if there exits some $\delta > 0$ such that, for each fluid solution with $\|X(0)\| \le 1$, one has $\|X(t)\| = 0$ for $t > \delta$. Under weak conditions, the stability of the fluid model implies the stability of the queueing network (e.g., the Harris recurrence of a Markov model in the case of renewal arrival processes, i.i.d. service times and Markov routing). See [16, 17] for a discussion of such results.

3.2.2 Utilization Condition

First we introduce some notation. Let V and W be matrices of size $L \times N$; then $V \preceq W$ means $v_{ij} \le w_{ij}$, for $i = 1, \ldots, L$ and $j = 1, \ldots, N$. Also, $\mathbf{1}_L$ and $\mathbf{0}_L$ stand for column vectors of 1's and 0's with length L, respectively and $\mathbf{0}_{L\times N}$ stands for the $L \times N$ matrix of all 0 entries.

Define the drift matrix D of the network by

$$D = M(R - I), \tag{3.2}$$

where $M = \mathrm{diag}\{\mu\}$. Since $Q := (I - R^T)^{-1}$ is a positive matrix, one has

$$D^{-T} = -M^{-1}Q \preceq \mathbf{0}_{K\times K}. \tag{3.3}$$

Furthermore, let the vector of nominal traffic of the queues be $\nu = [\nu_1, \nu_2, \ldots, \nu_K]^T$. We have

$$\nu_i = \lambda_i + \sum_{k=1}^{K} r_{ki}\nu_k.$$

In matrix form,

$$\nu = (I - R^T)^{-1}\lambda. \tag{3.4}$$

By Proposition 2.5.3. in [17], the utilization conditions of the network are

$$\sum_{i\in\mathcal{G}_j} \frac{\nu_i}{\mu_i} < 1, \quad \forall j = 1, \ldots J. \tag{3.5}$$

They express that work for each group arrives at the network at rate less than one.

Using (3.3) and (3.4), we find an equivalent stability conditions which is that $\tilde{\mathbf{e}}_j^T D^{-T}\lambda + 1 > 0$ for all $j = 1, \ldots, J$, where $\tilde{\mathbf{e}}_j$ is a column vector of length K corresponding to group j such that component i of $\tilde{\mathbf{e}}_j$ is $\tilde{e}_j(i) = 1_{\{i\in\mathcal{G}_j\}}$.

3.3 LQ Scheduling

In this section, we first provide a brief review of Filippov's theory for differential equations with discontinuous right-hand side. Then, we define LQ scheduling policy and analyze it in the fluid model. We show that LQ scheduling is throughput-optimal in the queueing network with two groups of two queues. We provide two different proofs. The first proof uses Filippov's theory to investigate the possible trajectories of the state and show that they can only go to zero. The second proof is based on finding a piece-wise linear Lyapunov function for the system.

3.3.1 Review of Filippov's Theory for Differential Equations with Discontinuous Right-Hand Side

In this subsection, we review Filippov's theory [26, 23] about solutions of differential equations with discontinuous right-hand side. In Subsection 3.3.2, we will see that the fluid limit equations of LQ scheduling are piece-wise smooth differential equations. We describe Filippov's theory in the following simple case. Let $h \in C^2 : \mathbb{R}^n \to \mathbb{R}$ be a function with continuous first and second derivatives. Let H_1 and H_2 be subspaces in $\mathbb{R}^n$, and let H_b be a hypersurface such that $\mathbb{R}^n = H_1 \cup H_2 \cup H_b$ is partitioned into H_1, H_2 and H_b. The hypersurface is defined by a function $h(x)$:

$$H_b = \{x \in \mathbb{R}^n | h(x) = 0\}.$$

Then, subspaces H_1 and H_2 are defined as

$$H_1 = \{x \in \mathbb{R}^n | h(x) < 0\},$$
$$H_2 = \{x \in \mathbb{R}^n | h(x) > 0\}.$$

Consider the nonlinear system

$$\dot{x} = f(x) = \begin{cases} f_1(x), & x \in H_1, \\ f_2(x), & x \in H_2. \end{cases}$$

Assume that the system is piece-wise smooth in the sense that $f_i(x) \in C^1$ is continuously differentiable in H_i, but does not extend smoothly over the boundary. The function $f(x)$ is not defined for $x \in H_b$. Filippov's theory states that under some mild conditions the solution to the differential equation is unique and $f(x)$ at the boundary is uniquely determined by a convex combination of f_1 and f_2:

$$f(x) = (1 - p(x))f_1(x) + p(x)f_2(x), \quad p \in [0, 1], \ x \in H_b.$$

This is known as *Filippov convex method.* Let $n(x)$ be the normal to the hypersurface H_b directing towards H_1. Suppose that $f_i(x) \in C^1$, $h(x) \in C^2$. The solution to the differential equation is unique in 2 of the following 3 cases. In case 1, at $x \in H_b$, $(n^T(x)f_1(x))(n^T(x)f_2(x)) > 0$. Then, x leaves H_b and enters H_1 if $n^T(x)f_1(x) > 0$, and enters H_2 if $n^T(x)f_1(x) < 0$. The intuitive reason behind the uniqueness of solution in this case is that both vector fields, f_1 and f_2, direct towards one subspace. This is known as *transversal intersection.* In case 2, $(n^T(x)f_1(x))(n^T(x)f_2(x)) < 0$ and $n^T(x)f_1(x) < 0$. Then, we have a so-called *attractive sliding mode* through x. Furthermore,

$$p(x) = \frac{n^T(x)f_1(x)}{n^T(x)(f_1(x) - f_2(x))}.$$

The intuitive reason behind the uniqueness of solution in this case is that when the trajectory is in a neighborhood of x, both vector fields direct towards the boundary hypersurface. In case 3, $(n^T(x)f_1(x))(n^T(x)f_2(x)) < 0$ and $n^T(x)f_1(x) > 0$. This is called a *repulsive sliding mode* and does not lead to uniqueness in solution. However, the repulsive sliding mode cannot occur in real physical systems [26]. The intuitive reason behind this is that even though in theory there exists a convex combination of the vector fields which lead to sliding of the trajectory on the hypersurface, the sliding mode is unstable in the sense that if a solution is in a neighborhood of x it will enter one of the hypersurfaces without crossing the boundary. Finally, if $n^T(x)f_1(x) = 0$ ($n^T(x)f_2(x) = 0$), then $p(x) = 0$ ($p(x) = 1$). Figure 3.3 illustrates Filippov's theory.

3.3.2 LQ Scheduling for Multiclass Queueing Networks

In this subsection, we provide the fluid model equations for LQ scheduling. Under this discipline, the longest queue in each group is served. To break ties, we can use a static

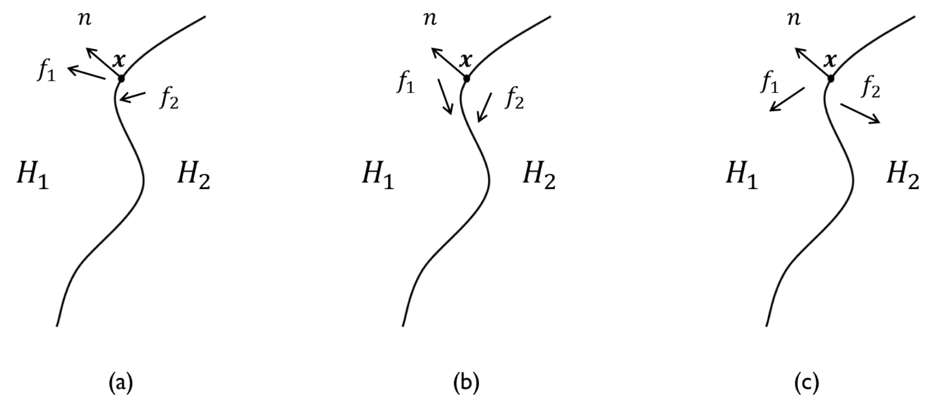

Figure 3.3: Illustration of Filippov's theory: (a) transversal intersection: $n^T f_1$ and $n^T f_2$ have the same sign, and the trajectory crosses the intersection and enters H_1; (b) attractive sliding mode: $n^T f_1 < 0$ and $n^T f_2 > 0$, and the trajectory slides on the boundary; (c) repulsive sliding mode: $n^T f_1 > 0$ and $n^T f_2 < 0$. In this case, the solution is not unique. The trajectory can slide on the boundary, or enters any of the subspaces. The reason that the sliding mode is not attractive is that a solution which starts in a neighborhood of x leaves the intersection.

priority scheduling among the maximum queues, or serve any of the maximum-length queues randomly. As we will see, the fluid model equations of the queueing network under LQ scheduling is a piece-wise affine dynamical system. More precisely, one has $\dot{X} = a$ if $X \in S$, where a is a constant drift vector and S is a subspace of $\mathbb{R}_+^K$ which is specified by the set of longest queues in each group. We define different states for the dynamical system which corresponds to different subspaces in which the drift vector is uniquely defined. Furthermore, we prove Lemma 11 that explicitly determines this constant drift vector for different states of the system.

Let $S_j(t)$ be the set of queues with maximum length in group j and $|S_j(t)| = L_j$ at time t. If all the queues in group $\mathcal{G}_j$ are empty[1], we use the convention that $S_j = \emptyset$. Define $S(t) = \cup_j S_j(t)$ and $|S(t)| = L = \sum_j L_j$. LQ scheduling corresponds to the fluid limit defined by (3.1) together with

$$\sum_{i \in S_j} \dot{T}_i(t) = 1$$
$$\sum_{i \in \mathcal{G}_j \backslash S_j} \dot{T}_i(t) = 0, \tag{3.6}$$

which states that the server in each group spends all its time serving the longest queues. The main result of this chapter is the following theorem.

[1]By empty queue in the fluid model, we mean a zero fluid level queue.

Theorem 8. *The multiclass queueing network in Figure 3.2, with two groups of two queues, is stable under LQ scheduling if the utilization conditions stated in* (3.5) *hold.*

The rest of this section establishes the proof of Theorem 8. Note that the stability of multiclass queueing networks with more than two groups of two queues under LQ scheduling is still an open question.

Define the function $f(t)$ as follows.

$$f : [0, \infty) \to \mathbb{R}^J, \; f(t) = [\max_i \{X_i\}_{i \in \mathcal{G}_1}, \ldots, \max_i \{X_i\}_{i \in \mathcal{G}_J}]^T. \tag{3.7}$$

Then, $\dot{f}(t)$ is the vector of drifts of maximum-length queues in the J groups.

An outline of the proof for stability of LQ scheduling in multiclass queueing networks with two groups of two queues is as follows. The drifts of queues with maximum length in one group are equal at a regular point by Lemma 9. A regular point is a value of $t \in [0, \infty)$ at which $f(t)$ in (3.7) is differentiable in all of its components. In Lemma 11, we show that the vector of drifts of maximum-length queues in different groups is strictly negative in at least one of its components. More precisely, if t is a regular point, $\dot{f}(t)$ has a constant component i determined by $S(t)$ such that $\dot{f}_i(t) \leq -\epsilon(S(t))$ for some $\epsilon(S(t)) > 0$ if $|X(t)| > 0$. This implies that $S(t)$ cannot remain constant as $t \to \infty$ unless $X(t) = \mathbf{0}$ (See Lemma 12). However, this is not enough to prove stability, as $S(t)$ can in principle travel around different states forever. We will show that there is no loop in the trajectory of $S(t)$, and this implies that $S(t)$ reaches the state in which all the queues are empty, and $S(t)$ remains in the absorbing "*zero*" state. A formal proof of this claim is given in Lemma 15. In this state, $\dot{T}_i = \nu_i/\mu_i$ is acceptable by Equation (3.5) and leads to $\dot{X}_i = 0, \; \forall i, \; 1 \leq i \leq K$. Since the zero state is feasible, $S(t)$ remains in this state for a positive amount of time. Now since no new maximum will appear in the network in the zero state, the fluid limit equations will not change, and $S(t)$ remains in this absorbing state forever.

Note that $S(t)$ goes from one state to another when a new queue appears in the set of maxima in one group. If in state S, $|S_j| \leq 1$ for all j, the groups are either empty or have a unique maximum-length queue. Then, the right-hand side of the differential equation is uniquely determined, and can be easily calculated. If there are multiple maxima in one or more groups, the state corresponds to the boundary of two subspaces, and whether this state is feasible (the trajectory slides on this boundary) or the state is not feasible (the trajectory leaves the boundary and enters one subspace) can be determined by Filippov's theory. This will be clarified in the following example.

Example: Consider the Lu-Kumar network (Figure 1) in the fluid limit. For this network, the discontinuity of the right-hand side of the differential equation is on two hyperplanes: $\Sigma_1 = \{X : X_1 = X_2\}$, $\Sigma_2 = \{X : X_3 = X_4\}$. The right-hand side of the differential equation has no ambiguity in the following subspaces: $H_1 = \{X : X_1 > X_2, X_3 > X_4\}$ which implies $\dot{X} = f_1 = [\lambda - \mu_1, 0, \mu_1 - \mu_3, \mu_3]$, $H_2 = \{X : X_1 < X_2, X_3 > X_4\}$ which implies $\dot{X} = f_2 = [\lambda, -\mu_2, -\mu_3, \mu_3]$, $H_3 = \{X : X_1 > X_2, X_3 < X_4\}$ which implies $\dot{X} = f_3 = [\lambda - \mu_1, \mu_4, \mu_1, -\mu_4]$, and $H_4 = \{X : X_1 < X_2, X_3 < X_4\}$ which implies $\dot{X} = f_4 = [\lambda, \mu_4 - \mu_2, 0, -\mu_4]$. Now let $H_b = \{X : X_1 = X_2, X_4 > X_3\} \subset \Sigma_1$ be the boundary of H_3 and H_4. Then, $n = [1, -1, 0, 0]^T$. Using Filippov theory, we have

$$n^T f_3 = \lambda - \mu_1 - \mu_4 < 0,$$

by utilization condition, and

$$n^T f_4 = \lambda - \mu_4 + \mu_2.$$

Thus, if $\mu_4 \leq \mu_2 + \lambda$, then we have a sliding mode, or equivalently a feasible state $(1, 2, 4)$. If $\mu_4 > \mu_2 + \lambda$, then we have a transversal intersection. Equivalently, state $(1, 2, 4)$ is not feasible and X crosses the boundary and goes to H_4 or state $(2, 4)$.

In the queueing network setting, Filippov's result is equivalent to server 1 using time-sharing to schedule queues 1 and 2 and keeping them equal. The time-sharing equation is

$$\lambda - \mu_1 \dot{T}_1 = \mu_4 - \mu_2(1 - \dot{T}_1).$$

Thus, state $(1, 2, 4)$ is feasible if $0 \leq \dot{T}_1 \leq 1$, since a feasible time-sharing can keep the queues 1 and 2 equal. Solving the time-sharing equation for $\dot{T}_1$ leads to the same feasibility condition that $\mu_4 \leq \mu_2 + \lambda$. Note that the convex combination of f_1 and f_2 is exactly server 1 using time-sharing to schedule queues 1 and 2. Now assume that $X(t)$ first lies in H_3, and then hits the boundary H_b (state $(1, 2, 4)$). Below, we will show the two possible transitions for the sliding mode and transversal intersection as

$$(1, 4) \to (1, 2, 4) \text{ and } (1, 4) \to (1, 2, 4)_0 \to (2, 4),$$

respectively, where $(1, 2, 4)_0$ indicates that $(1, 2, 4)$ is not feasible, so it is not visited for a positive amount of time.

Remark 4. Note that in general for two or more sliding surfaces, Filippov theory might not lead to uniqueness. For example consider two hypersurfaces $h_1(x) = 0$ and $h_2(x) = 0$ and 4 subspaces that are trivially defined by $h_i(x) = 0,\ i = 1, 2$. Let $f_j,\ j = 1, 2, 3, 4$ be the right-hand side of differential equations for the 4 subspaces. Then, in the boundary $h_1(x) = 0$ and $h_2(x) = 0$, in general one has $f(x) = \sum_{j=1}^{4} p_j f_j(x)$ and $\sum_j p_j = 1$. To have a sliding motion, one needs to have $n_1^T f = 0$ and $n_2^T f = 0$. These equations give 3 equations and 4 unknowns. However, in a queueing network the convex combination is constrained by the capacity of each server. That is $\sum_{i \in \mathcal{G}_j} \dot{T}_i = 1$. The extra equation is enough to lead to a unique solution.

As mentioned, LQ scheduling leads to a piece-wise affine dynamical system. Thus, when $X(t)$ lies in the interior of a subspace of $\mathbb{R}_+^n$ defined by the maxima in each group or equivalently in one state, t is regular, as the drift vector is a constant uniquely defined. The only possible non-regular points of the system are times t that $X(t)$

hits a boundary of subspaces or equivalently changes state. Therefore, if S_t remains constant in $[t, t+\epsilon]$ for some $\epsilon > 0$, then t is a regular point as $f(t)$ is differentiable at t.

Now we provide two Lemmas from [20] with a short proof sketch.

Lemma 9. *At a regular point t, $\dot{X}_i(t) = \dot{X}_j(t)$ if i and j are in the set of maxima of same group.*

Proof. Since i and j are in the same group and both in the set of maxima, $X_i = X_j$. Thus, if t is a regular point, X_i and X_j are differentiable and $\dot{X}_i(t) = \dot{X}_j(t)$. ■

Lemma 10. *Consider one arbitrary queue i and assume that t is a regular point such that $X_i(t) = 0$. Let ν_{in} be the total arrival rate and ν_{out} be the departure rate of queue i at time t. Then $\nu_{in} = \nu_{out}$.*

Proof. Since t is regular and $X_i(t) = 0$, X_i is differentiable at t and $\dot{X}_i(t) = 0$. This proves that $\nu_{in} = \nu_{out}$. ■

Lemma 11. *Suppose that the network is non-empty; that is, $\max_i X_i(t) > 0$. At a regular point t, $\alpha \triangleq \dot{f}(t)$ cannot be a non-negative vector. Furthermore, at each state S, the drift vector α is uniquely determined by the state S, and is independent of the queue-lengths.*

Proof. First, for ease of notation we prove the lemma for the case that all the groups are non-empty. Define matrix $E_{L\times J}$ as the following.

$$E = [e_1, e_2, \ldots, e_J] \tag{3.8}$$

where e_j is the column vector of length L defined as follows:

$$e_j = [\mathbf{0}_{\sum_{k=1}^{j-1} L_k}, \mathbf{1}_{L_j}, \mathbf{0}_{\sum_{k=j+1}^{J} L_k}]^T. \tag{3.9}$$

Recall that $L = |S|$ and $L_j = |S_j|$. Let $D_L = [d_{ik}]_{i,k\in S}$, $\dot{T}_L = [\dot{T}_i]_{i\in S}$ and $\lambda_L = [\lambda_i]_{i\in S}$ be the corresponding drift matrix, time-sharing vector and arrival vector to S, the set of maximum length queues. By the fluid model equations (3.1) and Lemma 9, we have the following matrix equation:

$$\begin{bmatrix} -(D_L)^T & E \\ E^T & \mathbf{0}_{J\times J} \end{bmatrix} \begin{bmatrix} \dot{T}_L \\ \alpha \end{bmatrix} = \begin{bmatrix} \lambda_L \\ \mathbf{1}_J \end{bmatrix}. \tag{3.10}$$

This is a simple representation of Equations (3.1) and (3.6) in matrix form which will be clarified in the following example.

Example: Consider the Lu-Kumar network in Figure 1 and $S = (1, 2, 4)$. Then,

$$D_L = \begin{pmatrix} -\mu_1 & 0 & 0 \\ 0 & -\mu_2 & 0 \\ 0 & \mu_4 & -\mu_4 \end{pmatrix},$$

$\lambda_L = [\lambda, 0, 0]^T$, and $\dot{T}_L = [\dot{T}_1, \dot{T}_2, \dot{T}_4]^T$. The matrix equation $E^T \dot{T}_L + \mathbf{0} \times \alpha = \mathbf{1}_J$ is exactly Equation 3.6 for the two groups, and $-(D_L)^T \dot{T}_L + E\alpha = \lambda_L$ is exactly Equation 3.1.

Solving Equation (3.10) using the block inverse formula, we have

$$\alpha = (E^T (D_L)^{-T} E)^{-1} \left(E^T (D_L)^{-T} \lambda_L + \mathbf{1}_J \right). \tag{3.11}$$

A careful observation shows that $E^T (D_L)^{-T} \lambda_L + \mathbf{1}_J$ is a positive vector, by the stability condition of these subset of queues which is a weaker condition than the stability condition of the whole network.

In Section 3.2.2, we showed that D^{-T} is a negative matrix. D_L has the same properties as D so, $E^T (D_L)^{-T} E$ is also a negative matrix. Now since $E^T (D_L)^{-T} E\alpha$ is a positive vector, α cannot be a non-negative vector. Furthermore, by longest-queue scheduling, the time-sharing vector $\dot{T}_L$ is a constant only depending on S; thus, $\alpha(S)$ is independent of the queue sizes.

If some groups are empty, the network is reduced to a smaller network for which the above proof still holds, since the utilization condition for the subset of non-empty groups is weaker than the utilization condition of the whole network. For instance, if group 2 in Lu-Kumar network of Figure 3.1 is empty, the network reduces to two queues (1 and 2) in series, and one can similarly show that for this sub-network, the constant drift vector α is non-negative. ■

Lemma 11 shows that some non-zero maxima must decrease, so that $S(t)$ cannot remain in any state but the zero state which is the only absorbing state.

Lemma 12. *There do not exist a non-zero state S and some $t_0 \geq 0$ such that $S(t) = S$ for all $t \geq t_0$.*

Proof. Suppose that $S(t) = S$ for all $t \geq t_0$ and S is non-empty. Then, by Lemma 11, there exists some j such that for $t \geq t_0$, $\alpha_j(t) = \dot{f}_j(t) = -\epsilon$ for some $\epsilon > 0$. This implies that the maximum queue-length in group j goes to $-\infty$ as $t \to \infty$, which is a contradiction. ■

Lemma 13. *Suppose that $S(t)$ switches from a feasible state $S^{(1)}$ to a feasible state $S^{(2)}$ at time t_0; that is, there exists an arbitrarily small $\epsilon > 0$, $S(t_0 - \epsilon) = S^{(1)}$ and $S(t_0 + \epsilon) = S^{(2)}$. Let $S_\Delta = S^{(1)} \setminus S^{(2)}$. Then, if $i \in S_\Delta$ and $i \in \mathcal{G}_j$, the maxima of $\mathcal{G}_j$ have positive drift in state $S^{(2)}$, unless $S_j^{(2)} = S^{(2)} \cap \mathcal{G}_j = \emptyset$.*

Proof. If $i \in S_\Delta$ and $S_j^{(2)} \neq \emptyset$, queue i is removed from the set of maxima of group j after transition to state $S^{(2)}$. This requires that, in state $S^{(2)}$, queue i has a smaller drift than the other maxima in its group. To see this formally, let $k \in S_j^{(2)}$. Let $t_1 = \inf\{t > t_0 : \ S(t) \neq S^{(2)}\}$. Then, $S(t) = S^{(2)}$ for $t \in (t_0, t_1)$ for some $t_1 > t_0$ since $S^{(2)}$ is a feasible state by assumption. Then, $X_i(t_0) = X_k(t_0)$ by continuity of fluid.

Now, suppose that $\dot{X}_i(t) \geq \dot{X}_k(t)$ in the interval $t \in (t_0, t_0+\epsilon)$ for some $0 < \epsilon < t_1 - t_0$. This contradicts the assumption that $i \notin S^{(2)}$. Thus, $\dot{X}_i(t) < \dot{X}_k(t), \forall k \in S_j^{(2)}$ for $t \in (t_0, t_1)$. This is due to the fact that in every state the queue-length drifts are constants that are uniquely determined by that state, so $\dot{X}_i(t)$ is a constant independent of t for $t \in (t_0, t_1)$ for all i. Now, since queue i is not served, one has $\dot{X}_i(t) \geq 0$ for $t \in (t_0, t_1)$. Consequently, the drift of the maxima in group j is positive in state $S^{(2)}$. ∎

We are now ready to prove the following result.

Lemma 14. *Under the stability conditions, there is no loop in the trajectories of the state diagram of the network with two groups of two queues.*

Proof. We prove the lemma by contradiction. To have a loop in the state diagram, there are three possible scenarios.

Scenario I: There is an empty group in one of the states in the loop.

Without loss of generality, we can consider this state to be $(1, \emptyset)$ or $(1, 2, \emptyset)$.

To show that this is not possible, we first show that $(1, 2, \emptyset)$ cannot be feasible. To see this, assume that $(1, 2, \emptyset)$ is feasible. If $(1, 2, \emptyset)$ is feasible, then group 1 has negative drift by Lemma 11. Thus, next $S(t)$ visits the absorbing $(\emptyset, \emptyset)$, which contradicts the existence of the loop. Hence, $(1, 2, \emptyset)$ is not feasible and $(1, \emptyset)$ must be the feasible state with an empty group in the loop.

Second, we claim that $(1, 2, 3, 4)$ cannot be feasible. Assume that it is. One then has the following transitions starting from $(1, \emptyset)$:

$$(1, \emptyset) \to (1, 2, \emptyset)_0 \to (1, 2, 3, 4).$$

In $(1, 2, 3, 4)$, the drift of group 2 is positive since it was previously empty. So queues 3 and 4 are necessarily increasing. Thus, group 1 has negative drift and next becomes empty. We claim that the next feasible state is then $(\emptyset, 3, 4)$. Indeed, if that state is not feasible, the next feasible state would then be $(\emptyset, 3)$ (without loss of generality). But, in that state, queue 3 must decrease (since group 1 has a zero drift and one of the groups must have a negative drift, by Lemma 11), and the other queue 4 does not decrease, which is not possible. Thus, the next transitions must be

$$(1, 2, 3, 4) \to (\emptyset, 3, 4) \to (\emptyset, \emptyset),$$

which contradicts the existence of a loop.

Third, we show that $S(t)$ must go to the zero state. Indeed, we have the following transitions:

$$(1, \emptyset) \to (1, 2, \emptyset)_0 \to (1, 2, 3, 4)_0.$$

The next state is either $(1, 2, 3)$ or $(1, 2, 4)$. To see this, first note that if one of queues 1 or 2 are removed from the set of maxima, by Lemma 13, group 1 has positive drift in the next state. On the other hand, since group 2 was previously empty, it necessarily has a positive drift in the next state. Since both of the groups cannot have a positive drift, this is a contradiction and it follows that 1 and 2 remain in the set of maxima, and group 1 has negative drift. Therefore, without loss of generality we can consider the next transition to be to $(1, 2, 3)$, i.e., we have the following transitions:

$$(1, \emptyset) \rightarrow (1, 2, \emptyset)_0 \rightarrow (1, 2, 3, 4)_0 \rightarrow (1, 2, 3). \tag{3.12}$$

Next, group 1 becomes empty and $S(t)$ visits $(\emptyset, 3)$. After some time, queues 3 and 4 become equal, as 3 has negative drift, so $S(t)$ visits $(\emptyset, 3, 4)$. We claim that $(\emptyset, 3, 4)$ must be feasible. If this state is non-feasible, $S(t)$ visits $(1, 2, 3, 4)_0$. Thus, the next transition is to $(1, 2, 3)$ by (3.12). However, this is not possible since in this state, group 1 has negative drift. Since group 1 is empty in $(\emptyset, 3, 4)$, it cannot have a negative drift in the next state. This shows that $(\emptyset, 3, 4)$ is feasible. Since $(\emptyset, 3, 4)$ is feasible, group 2 has negative drift by Lemma 11 and the next state is the zero state, which contradicts the existence of a loop.

Scenario II: None of the states in the loop has an empty group and there is a state of cardinality 3 in the loop. Note that a feasible state $(1, 2, 3, 4)$ cannot be in the loop since from that state one of the groups becomes empty. Without loss of generality suppose that this state is $S^{(1)} = (1, 2, 3)$ and the next transition happens when queue 4 appears in the set of maxima.

Case 1: The drift of queue 3 at $S^{(1)}$ is positive, so necessarily drifts of 1 and 2 are equal and negative by Lemma 11. Therefore, the state can only go to $S^{(2)} = (1, 2, 4)$ with drift of 4 positive and drifts of 1 and 2 negative by Lemma 13. From this point, no new maximum can appear until group 1 goes empty since $\dot{X}_4 > \dot{X}_3 > 0$ by Lemma 13 and we are back to Scenario I.

Case 2: The drift of queue 3 at $S^{(1)}$ is negative. The next state cannot be $(1, 2, 3, 4)$, so $S(t)$ can either go to $(1, 2, 4)$ with $\dot{X}_4 > 0$ in which we are in the same situation as Case 1, or $\dot{X}_3 = \dot{X}_4 < 0$, and one of the queues 1 or 2 (for example queue 2) is removed from the set of maxima. Note that by Lemma 13, queues 3 and 4 remain equal while having negative drift. From this point, no new maximum can appear until group 2 goes empty since $\dot{X}_1 > \dot{X}_2 > 0$ and we are back to Scenario I.

Scenario III: All of the states in the loop are of cardinality 2. Obviously, not all the maxima of a state can change at time t. So for example from $S = (1, 3)$, the next state cannot be $S = (2, 4)$. Now we consider 3 cases:

Case 1: There is a loop of length 2. Without loss of generality we can consider the loop to be $(1, 3) \rightarrow (1, 4) \rightarrow (1, 3)$. But at state $(1, 4)$, it is not possible that queue 3 appears again as the new maximum since $\dot{X}_4 > \dot{X}_3 > 0$. So there is no loop of length 2 in the state diagram.

Case 2: There is a loop of length 3. Without loss of generality we can consider the loop to be $(1, 3) \rightarrow (1, 4) \rightarrow (2, 4) \rightarrow (1, 3)$. But clearly the last transition in the loop is not valid. So there is no loop of length 3.

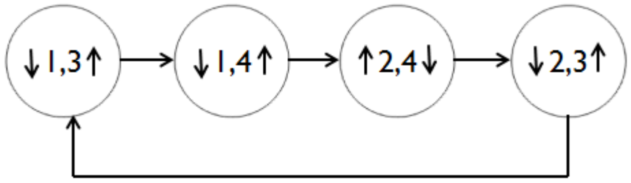

Figure 3.4: Loop in the state diagram

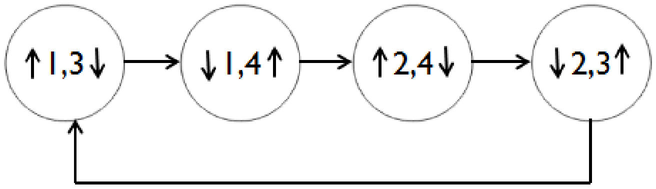

Figure 3.5: Loop in the state diagram

Case 3: There is a loop of length 4. Without loss of generality we can consider the loop to be $(1,3) \to (1,4) \to (2,4) \to (2,3) \to (1,3)$. We cannot rule out this case just by inspection like the previous cases. Note that in each of these states the drift of one group is positive and the other one negative $S(t)$ cannot go to a state of cardinality where one maximum is removed and both of the drifts are negative by Lemma 13. A careful observation shows that the sign of the drifts of each group should change in every jump. If not, as we can see in Figure 3.4, the last jump $(2,3) \to (1,3)$ is not valid by Lemma 13.

The last possible case that we have to investigate is the loop with the drifts shown in Figure 3.5.

The possibility of this loop happening cannot be rejected just by inspection. We write the exact conditions leading to each transition which is an inequality; then sum them up and reach a contradiction. The transition $(\uparrow 1,3 \downarrow) \to (\downarrow 1,4 \uparrow)$ implies that state $S = (1,3,4)$ is not feasible. Indeed solving equation $\dot{X}_3 = \dot{X}_4$, where

$$\dot{X}_3 = \lambda_3 - \mu_3\dot{T}_3 + \mu_1 r_{13} + \mu_3\dot{T}_3 r_{33} + \mu_4(1-\dot{T}_3)r_{43}$$
$$\dot{X}_4 = \lambda_4 - \mu_4(1-\dot{T}_3) + \mu_1 r_{14} + \mu_3\dot{T}_3 r_{34} + \mu_4(1-\dot{T}_3)r_{44},$$

leads to $\dot{T}_3 < 0$ which is not acceptable. So for $\dot{T}_3 = 0$, $\dot{X}_3 < \dot{X}_4$ and the following condition is obtained.

$$\lambda_4 - \mu_4 + \mu_1 r_{14} - \lambda_3 - \mu_1 r_{13} - \mu_4 r_{43} + \mu_4 r_{44} > 0. \tag{3.13}$$

Similarly, for the other 3 transitions we obtain 3 inequalities which are the following:

$$\lambda_2 - \mu_2 + \mu_4 r_{42} - \lambda_1 - \mu_4 r_{41} - \mu_2 r_{21} + \mu_2 r_{22} > 0 \tag{3.14}$$
$$\lambda_3 - \mu_3 + \mu_2 r_{23} - \lambda_4 - \mu_2 r_{24} - \mu_3 r_{34} + \mu_3 r_{33} > 0 \tag{3.15}$$
$$\lambda_1 - \mu_1 + \mu_3 r_{13} - \lambda_2 - \mu_3 r_{32} - \mu_1 r_{12} + \mu_1 r_{11} > 0. \tag{3.16}$$

Adding inequalities (3.13) to (3.16), we have

$$\begin{aligned} &-\mu_1(1 - r_{11} + r_{12} + r_{13} - r_{14}) - \mu_2(1 + r_{21} - r_{22} + r_{23} - r_{24}) \\ &-\mu_3(1 - r_{31} + r_{32} - r_{33} + r_{34}) - \mu_4(1 + r_{41} - r_{42} + r_{43} - r_{44}) > 0, \end{aligned}$$

which is clearly not true so we reach a contradiction and the proof of the lemma is complete.

■

The immediate result of Lemma 14 is that $S(t)$ will reach the zero state after a finite time since $S(t)$ cannot travel around a finite state diagram without making a loop and reaching the zero state.

Lemma 15. *Suppose that $S(t)$ reaches the zero state from a non-zero state. Then, $S(t)$ remains in the absorbing zero state.*

Proof. Without loss of generality, suppose that the zero state is reached from the feasible state $(1, 2, \emptyset)$. Note that one can have a case that the zero state is reached from the feasible state $(1, 2, 3, 4)$ if the initial condition and the network is designed such that both groups have negative drift in $(1, 2, 3, 4)$ and the queues become empty exactly at the same time. Nevertheless, the following proof also holds for this rare case. By Lemma 11, group 1 has negative drift in state $(1, 2, \emptyset)$. Thus, $S(t)$ visits the set $(\emptyset, \emptyset)_0$. If this state is feasible, there will be no change in the dynamics of the system so it is absorbing. In this case, $\dot{T}_i = \frac{\nu_i}{\mu_i}$. In contrary, suppose that $(\emptyset, \emptyset)$ is not feasible. Then, $S(t)$ visits $(1, 2, 3, 4)_0$ or $(\emptyset, 3, 4)_0$ at time t_0. By Lemma 11, at least one of the groups has negative drift in these states (if they are feasible) or the next *non-empty* feasible state. But this is not possible by continuity and non-negativity of the fluid since $X_i(t) = 0,\ \forall i$. To see this, without loss of generality assume that group 1 has negative drift at the next feasible state, and queue 1 belongs to the maxima of group 1. Then, for arbitrarily small $\epsilon > 0$, we have $\dot{X}(t_0 + \epsilon) < 0$ while $X(t_0) = 0$, which is not possible. ■

Remark 5. Note that in Lu-Kumar network (Figure 1) with priority scheduling and priorities given to states 2 and 3, the zero state is not feasible. First, it is not true that the zero state will be reached from a non-zero state. Indeed, it is shown that if the

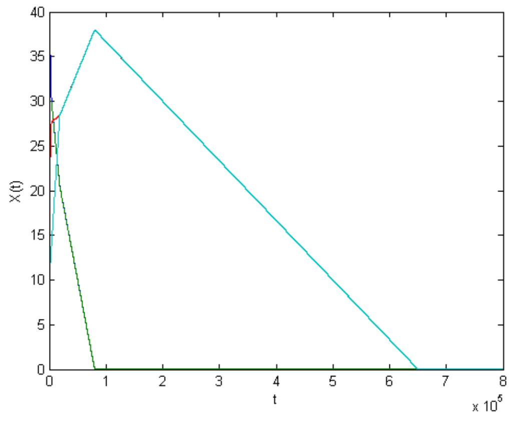

Figure 3.6: Trajectories of fluid in Lu-Kumar network

initial fluid level is non-zero, the zero state is never reached. Second, since queues 2 and 3 are never served at the same time, there is an extra condition that $\dot{T}_2 + \dot{T}_3 \leq 1$. In words, queues 2 and 3 form a virtual group. Thus, $\dot{T}_i = \frac{\nu_i}{\mu_i}$ can be non-feasible in general.

We show the simulation results for the Lu-Kumar network (Figure 3.1) in the fluid limit. In the simulation, service rates are $\mu = [3, 1, 1, 1]^T$, and arrival rate is $\lambda = 0.4$. Initial queue-lengths is $X(0) = [40, 30, 20, 10]^T$. Figure 3.6 shows the trajectories of the fluid in different queues versus time. As we can see, first the queue-lengths in different groups become equal. Then group 1 goes empty, and from that point drift of group 2 is negative until it is empty.

3.3.3 Second Proof of Theorem 8

The key idea for the second proof is to find a piece-wise linear Lyapunov function. Let $W(t) = M^{-1}QX(t) = -D^{-T}X(t)$. Note that $QX(t)$ denotes the vector of potential fluid level for each queue at time t, and $W_i(t)$ is the potential work to be done at queue i at time t. Let

$$a = \frac{\mu_1^{-1}(q_{13} + q_{14}) + \mu_2^{-1}(q_{23} + q_{24})}{\mu_3^{-1}(q_{33} + q_{34}) + \mu_4^{-1}(q_{43} + q_{44})}, \tag{3.17}$$

and

$$b = \frac{\mu_1^{-1}(q_{11} + q_{12}) + \mu_2^{-1}(q_{21} + q_{22})}{\mu_3^{-1}(q_{31} + q_{32}) + \mu_4^{-1}(q_{41} + q_{42})}, \tag{3.18}$$

where $Q = [q_{ij}]$. In Lemma 16, we show that $a \leq b$. Let $\beta \in [a, b]$. We show that

$$V(X) = \max(f_1, f_2), \tag{3.19}$$

where $f_1 = W_1 + W_2 = \tilde{\mathbf{e}}_1^T W$ and $f_2 = \beta[W_3 + W_4] = \beta\tilde{\mathbf{e}}_2^T W$, is a Lyapunov function.

Lemma 16. *The following inequality holds:*

$$a \leq b, \tag{3.20}$$

where a is given in (3.17) *and b is given in* (3.18)*.*

Proof. It is enough to show the following 4 inequalities:

$$(q_{11} + q_{12})(q_{33} + q_{34}) \geq (q_{13} + q_{14})(q_{31} + q_{32}) \tag{3.21}$$
$$(q_{11} + q_{12})(q_{43} + q_{44}) \geq (q_{13} + q_{14})(q_{41} + q_{42}) \tag{3.22}$$
$$(q_{21} + q_{22})(q_{33} + q_{34}) \geq (q_{23} + q_{24})(q_{31} + q_{32}) \tag{3.23}$$
$$(q_{21} + q_{22})(q_{43} + q_{44}) \geq (q_{23} + q_{24})(q_{41} + q_{42}). \tag{3.24}$$

We prove (3.21). The proof of (3.22)–(3.24) is identical with change of indices.

Recall that $Q = (I - R^T)^{-1}$. $I - R$ is an M-matrix [61]. That is, $I - R$ has non-negative diagonal entries, non-positive off-diagonal entries, and the sum of entries of each row is non-negative. M-matrices have positive principal minors [61]; thus, $\det(I - R)$ is strictly positive, as the network is open. The following equality can be computed:

$$\det\left(\sqrt{\det(I-R)}\begin{bmatrix} q_{11}+q_{12} & q_{13}+q_{14} \\ q_{31}+q_{32} & q_{33}+q_{34} \end{bmatrix}\right)$$
$$= (1 + r_{21} - r_{22})(1 + r_{43} - r_{44}) + (r_{23} - r_{24})(r_{42} - r_{41}). \tag{3.25}$$

It is now sufficient to show that the right-hand side of the above equality is non-negative. This can be shown as follows.

$$(1 + r_{21} - r_{22})(1 + r_{43} - r_{44}) + (r_{23} - r_{24})(r_{42} - r_{41})$$
$$\geq (1 - r_{22})(1 - r_{44}) - (r_{23} + r_{24})(r_{42} + r_{41}) \geq 0.$$

The last inequality is due to the facts $1 - r_{22} \geq r_{23} + r_{24}$ and $1 - r_{44} \geq r_{42} + r_{41}$. This completes the proof of the lemma. ■

Now we show that $V(X)$ is a Lyapunov function. First, it is clear that $V(X) = \mathbf{0}$ if and only if $X = 0$. We show that at a regular point t, $\dot{V}(t) \leq -\delta$ if $X(t) \neq \mathbf{0}$ for some $\delta > 0$ by considering 3 different cases. A time t is regular if each component of $X(t)$ is differentiable at time t, and function $V(X(t))$ is differentiable at time t.

- Case 1: Suppose that $f_1(t) > 0$ and $f_2(t) > 0$. The fluid model equation in (3.1) can be written in vector form as

$$X(t) = X(0) + \lambda t + D^T T(t) \tag{3.26}$$

Then,

$$\dot{f}_1(t) = \tilde{\mathbf{e}}_1^T M^{-1} Q(\lambda + D^T \dot{T}(t)) = -\tilde{\mathbf{e}}_1^T D^{-T}\lambda - 1 \leq -\epsilon,$$

by the utilization condition and the fact that $\dot{T}_1 + \dot{T}_2 = 1$ since LQ scheduling is work-conserving. Similarly,

$$\dot{f}_2(t) = -\tilde{\mathbf{e}}_2^T D^{-T}\lambda - 1 \leq -\epsilon.$$

Thus, $\dot{f}(t) \leq -\epsilon$ in this case.

- Case 2: Suppose that group 1 is empty, and group 2 is non-empty. That is, $X_1(t) = X_2(t) = 0$ and $X(t) \neq \mathbf{0}$. Then, similar to case 1, one can show that $\dot{f}_2(t) \leq -\epsilon$. However, $\dot{f}_1(t)$ is not necessarily strictly negative. Thus, we need to show that at all regular points t in case 2, either $\dot{f}_1(t) < 0$ or $f_2(t) \geq f_1(t)$ (or both). Since $X_1(t) = X_2(t) = 0$, one can write $f_1(t) = c_1X_3(t) + c_2X_4(t)$ and $f_2(t) = c_3X_3(t) + c_4X_4(t)$ for some positive constants c_i, $i = 1, 2, 3, 4$. We re-write f_1 in the following two expressions.

$$f_1 = \frac{c_1}{c_3} f_2 + (c_2 - \frac{c_1c_4}{c_3})X_4 \tag{3.27}$$

$$f_1 = \frac{c_2}{c_4} f_2 + (c_1 - \frac{c_2c_3}{c_4})X_3. \tag{3.28}$$

Recall that under LQ scheduling, $\dot{T}_3 = 1$ and $\dot{T}_4 = 0$ if $X_3 > X_4$, which implies that $\dot{X}_3 < 0$ and $\dot{X}_4 \geq 0$ in case 2, since $\dot{f}_2 = c_3\dot{X}_3 + c_4\dot{X}_4 < 0$. Similarly, by LQ scheduling $X_4 > X_3$ implies that $\dot{T}_4 = 1$, $\dot{T}_3 = 0$, $\dot{X}_4 < 0$, and $\dot{X}_3 \geq 0$ in case 2. Now suppose that $c_1c_4 < c_2c_3$. Then, we expand f_1 according to (3.27). Then, $X_4 > X_3$ implies that $\dot{f}_1 < -\frac{c_1}{c_3}\epsilon$. We show that if $X_3 \geq X_4$, $f_2 \geq f_1$ in this case as follows. Fixing $X_4 \geq 0$, the function

$$\frac{f_2}{f_1} = \frac{c_3X_3 + c_4X_4}{c_1X_3 + c_2X_4}$$

is increasing in X_3 since $c_1c_4 < c_2c_3$ by assumption. Thus,

$$\inf_{X_3:\ X_4 \leq X_3} \frac{f_2}{f_1}$$

is achieved when $X_3 = X_4$. Therefore, it is enough to show that $f_2 \geq f_1$ when $X_3 = X_4$. When $X_1 = X_2 = 0$ and $X_3 = X_4$, we have

$$\frac{f_2}{f_1} = \frac{\beta[\mu_3^{-1}(q_{33} + q_{34}) + \mu_4^{-1}(q_{43} + q_{44})]}{\mu_1^{-1}(q_{13} + q_{14}) + \mu_2^{-1}(q_{23} + q_{24})} = \frac{\beta}{a} \geq 1. \tag{3.29}$$

In the above derivation, we used (3.17) and the fact that $\beta \in [a, b]$.

Similarly, suppose that $c_1c_4 > c_2c_3$. Then, we expand f_1 according to (3.28). Then, $X_3 > X_4$ implies that $\dot{f}_1 < -\frac{c_2}{c_4}\epsilon$. We show that if $X_4 \geq X_3$, $f_2 \geq f_1$ in this case as follows. Fixing $X_3 \geq 0$, the function

$$\frac{f_2}{f_1} = \frac{c_3X_3 + c_4X_4}{c_1X_3 + c_2X_4}$$

is increasing in X_4 since $c_1c_4 > c_2c_3$ by assumption. Then,

$$\inf_{X_4:\ X_3 \leq X_4} \frac{f_2}{f_1}$$

is achieved when $X_4 = X_3$. Therefore, it is enough to show that $f_2 \geq f_1$ when $X_3 = X_4$, which is already shown in (3.29).

Finally, supposing that $c_1c_4 = c_2c_3$, $f_1/f_2 = c_1/c_3$. Thus,

$$\dot{f}_1 = \frac{c_1}{c_3}\dot{f}_2 \leq -\frac{c_1}{c_3}\epsilon.$$

- Case 3: Suppose that group 2 is empty, and group 1 is non-empty. Then, $\dot{f}_1(t) \leq -\epsilon$. Similar to case 2, we need to show that either $\dot{f}_2(t) < 0$ or $f_1(t) \geq f_2(t)$ (or both). Since in case 3, $X_3 = X_4 = 0$, one can write $f_1 = c'_1X_1 + c'_2X_2$ and $f_2 = c'_3X_1 + c'_4X_2$ for some positive constants c'_i, $i = 1, 2, 3, 4$. We have

$$f_2 = \frac{c'_3}{c'_1}f_1 + (c'_4 - \frac{c'_3c'_2}{c'_1})X_2 \tag{3.30}$$

$$f_2 = \frac{c'_4}{c'_2}f_1 + (c'_3 - \frac{c'_4c'_1}{c'_2})X_1. \tag{3.31}$$

Similar to case 2, by LQ scheduling, $X_1 > X_2$ implies that $\dot{X}_1 < 0$ and $\dot{X}_2 \geq 0$. Also, $X_2 > X_1$ implies that $\dot{X}_2 < 0$ and $\dot{X}_1 \geq 0$. Now suppose that $c'_1c'_4 > c'_2c'_3$. Then, $X_2 > X_1$ implies that $\dot{f}_2 < -\frac{c'_3}{c'_1}\epsilon$ due to (3.30). We show that if $X_1 \geq X_2$, $f_1 \geq f_2$. Fixing $X_2 \geq 0$, the function

$$\frac{f_1}{f_2} = \frac{c'_1X_1 + c'_2X_2}{c'_3X_1 + c'_4X_2}$$

is increasing in X_1 since $c'_1c'_4 > c'_2c'_3$. Thus,

$$\inf_{X_1: X_1 \geq X_2} \frac{f_1}{f_2}$$

is achieved when $X_1 = X_2$. Therefore, it is enough to show that $f_1 \geq f_2$ when $X_1 = X_2$. When $X_3 = X_4 = 0$ and $X_1 = X_2$, we have

$$\frac{f_1}{f_2} = \frac{\mu_1^{-1}(q_{11} + q_{12}) + \mu_2^{-1}(q_{21} + q_{22})}{\beta[\mu_3^{-1}(q_{33} + q_{34}) + \mu_4^{-1}(q_{41} + q_{42})]} = \frac{b}{\beta} \geq 1. \tag{3.32}$$

In the above derivation, we used (3.18) and the fact that $\beta \in [a, b]$.

Similarly, suppose that $c'_1 c'_4 < c'_2 c'_3$. Then, $X_1 > X_2$ implies that $\dot{f}_2 < -\frac{c'_4}{c'_2}\epsilon$ due to (3.31). We show that if $X_2 \geq X_1$, $f_1 \geq f_2$. Fixing $X_1 \geq 0$, the function $\frac{f_1}{f_2}$ is increasing in X_2 since $c'_2 c'_3 > c'_1 c'_4$. Then,

$$\inf_{X_2:\ X_1 \leq X_2} \frac{f_1}{f_2}$$

is achieved when $X_2 = X_1$. Therefore, it is enough to show that $f_1 \geq f_2$ when $X_1 = X_2$, which is already shown in (3.32).

Finally, supposing that $c'_1 c'_4 = c'_2 c'_3$, $f_1/f_2 = c'_1/c'_3$. Thus,

$$\dot{f}_2 = \frac{c'_3}{c'_1}\dot{f}_1 \leq -\frac{c'_3}{c'_1}\epsilon.$$

Now taking

$$\delta = \min(\epsilon, \frac{c_1}{c_3}\epsilon, \frac{c_2}{c_4}\epsilon, \frac{c'_3}{c'_1}\epsilon, \frac{c'_4}{c'_2}\epsilon) > 0,$$

we have $\dot{V}(t) \leq -\delta$. This completes the proof of Theorem 8.

3.4 LDQ Scheduling

3.4.1 Policy

In each group, we define a queue to be "dominating" if it is belongs to the set of longest queues in the network, or if it cannot feed a longest queue in the network. Let $\bar{S}'$ be the set of global maxima in the network. For each group j, the dominating set $\bar{\mathcal{D}}_j$ is defined to be

$$\bar{\mathcal{D}}_j = \{i \in \mathcal{G}_j : r_{is} = 0 \text{ if } \bar{X}_i < \bar{X}_s, \forall s \in \bar{S}'\}.$$

The scheduling policy is to serve the longest queue in $\bar{\mathcal{D}}_j$. If $\bar{\mathcal{D}}_j = \emptyset$, do not serve any queues from group j. As an example, see Figure 3.7, where queue 3 is the maximum with length 30. Since jobs leaving both queues 1 and 2 can be destined to queue 3, $\bar{\mathcal{D}}_1 = \emptyset$ and $\bar{\mathcal{D}}_2 = \{3, 4\}$, so no queues in group 1 is served, and queue 3 in group 2 is served. As we can see, while the policy is not work-conserving, it always serves the queue with maximum-length in the network. The policy is robust to the knowledge of service rates and exact values of routing probabilities. The draw-back in comparison with LQ scheduling is that it needs global knowledge of queue lengths and the topology of the network.

The main result of this section is the following theorem.

Theorem 9. *A multiclass queueing networks is stable under LDQ scheduling if the utilization conditions stated in* (3.5) *hold and the network is acyclic.*

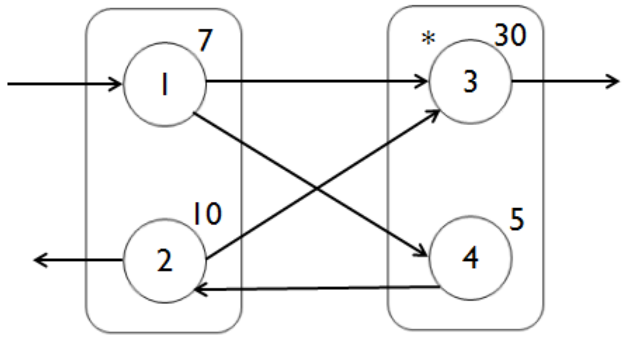

Figure 3.7: LDQ is not work-conserving

3.4.2 Proof of Theorem 9

We analyze the fluid model of the system under longest-dominating-queue scheduling policy. Let

$$S' = \{i : X_i = \max_k X_k\}$$

be the set of maximum-length (maximum fluid level) queues in the network, and let $|S'| = L'$. Let

$$\mathcal{D}_j = \{i \in \mathcal{G}_j : r_{is} = 0 \text{ if } X_i < X_s, \forall s \in S'\}$$

be the set of dominating queues in group j in the fluid model. Let

$$\tilde{S}_j = \{i : i \in \mathcal{D}_j, X_i = \max_{k \in \mathcal{D}_j} X_k\}$$

be the set of longest dominating queues in group j. Define a subset of groups $\mathcal{J} \subseteq \{1, 2, \ldots, J\}$ as follows:

$$\mathcal{J} = \{j | \exists i \in \mathcal{G}_j \cap S' \text{ such that } r_{is} = 0, \forall s \in S'\}.$$

In words, $\mathcal{J}$ is the set of groups that have a globally maximum-length queue which does not feed another globally maximum-length queue. The scheduling policy implies that

$$\sum_{i \in \tilde{S}_j} \dot{T}_i(t) = 1, \quad \text{if } j \in \mathcal{J}, \tag{3.33}$$

$$\sum_{i \in \tilde{S}_j} \dot{T}_i(t) \leq 1, \quad \text{if } j \notin \mathcal{J}, \tag{3.34}$$

$$\sum_{i \in \mathcal{G}_j \setminus \tilde{S}_j} \dot{T}_i(t) = 0. \tag{3.35}$$

The fluid model equations for this policy are the one stated in (3.1) together with (3.33)–(3.35).

We prove Theorem 9 using the Lyapunov function $V(X) = \max_i\{X_i\}$. Let $S'_j = \mathcal{G}_j \cap S'$ be the set of maximum-length (maximum fluid level) queues in group j. At time t, if $L' = 1$, that is the queue with maximum fluid level is unique, $\dot{V}(X(t))$ is clearly negative. Here is a proof. Let $S' = \{i\}$. Then, $\dot{T}_i = 1$. Moreover, no jobs will be scheduled destined to the queue with maximum length in the network (not necessarily the maximum in one group). Thus, $\dot{V}(X(t)) = \lambda_i - \mu_i < 0$. We prove that if the maximum is not unique, in a regular point the drift of the set of maxima are equal and negative. Suppose that $L' > 1$. In a regular point, by Lemma 9, the drifts of all the queues in S' are equal. Since there is no flow to the sub-network of queues in S' coming from other queues, we can only consider this sub-network to analyze the drift of the Lyapunov function. Let the corresponding drift matrix and arrival vectors to set S' be $D_{L'}$ and $\lambda_{L'}$. Suppose that J' groups ($J' \leq J$) have queues in S' so $S' = \cup_{j=1}^{J'} S'_j$ (with some abuse of notation due to relabelling the J' groups by 1 to J'). Let $L'_j = |S'_j|$. In a regular point t, the drift of the queues in this sub-network S' are all equal to $\dot{V}(t) = \alpha$.

Lemma 17. *If the network is non-empty,* $\dot{V}(t) = \alpha < 0$.

Proof. Define matrix $E_{L' \times J'}$ as the following.

$$E = [e_1, e_2, \ldots, e_{J'}], \tag{3.36}$$

where e_j is a column vector of length L'.

$$e_j = [\mathbf{0}_{\sum_{k=1}^{j-1} L'_k}, \mathbf{1}_{L'_j}, \mathbf{0}_{\sum_{k=j+1}^{J'} L'_k}]^T \tag{3.37}$$

The matrix equation will be

$$\begin{bmatrix} -D'_L & E \\ E^T & \mathbf{0}_{J' \times J'} \end{bmatrix} \begin{bmatrix} \dot{T}_{L'} \\ \alpha \mathbf{1}_{J'} \end{bmatrix} = \begin{bmatrix} \lambda_{L'} \\ t_{J'}, \end{bmatrix} \tag{3.38}$$

where vector $t_{J'} = [t_j]$, $t_j = \sum_{i \in S'_j} \dot{T}_i \leq 1$. By (3.33), at least one of the elements of $t_{J'}$ is equal to 1, if the network topology is acyclic. The reason is that the sub-network certainly has a flow that is leaving the sub-network via an "output" queue. Note that if there is a cycle in the network, such a queue may not exist. (See Figure 3.8) The group which contains the "output" queue, let's say group j^*, has the property that $j^* \in \mathcal{J}$; thus, $\sum_{i \in S'_{j^*}} \dot{T}_i = 1$.

Solving Equation (3.38) using block inverse formula, we have

$$\alpha \mathbf{1}_{J'} = (E^T D_{L'}{}^{-T} E)^{-1} \left(E^T D_{L'}{}^{-T} \lambda_{L'} + t_{J'} \right). \tag{3.39}$$

The vector $\left(E^T D_L{}^{-T} \lambda_{L'} + t_{J'} \right)$ is positive in at least one element corresponding to group j^* by the stability condition of this sub-network. (which is a weaker condition than the stability condition of the whole network) We have already proved that

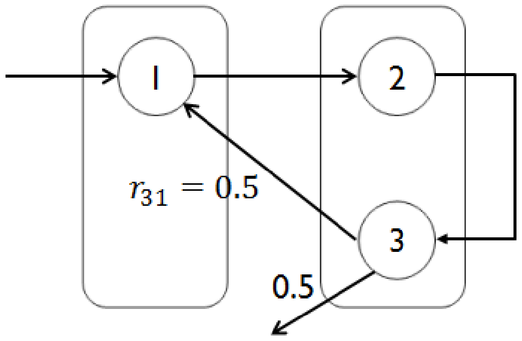

Figure 3.8: No output queue in a cycle

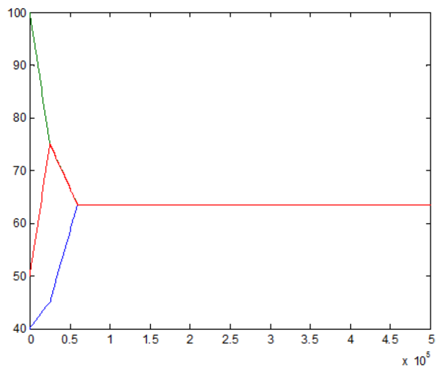

Figure 3.9: LDQ in a cycle

$E^T D_{L'}{}^{-T} E$ is a negative matrix. Now since $E^T D_{L'}{}^{-T} E(\alpha \mathbf{1}_{J'})$ is not a negative vector, α cannot be non-negative.

■

Consequently, $V(X)$ is a Lyapunov function and proof of Theorem 9 is complete.

The simulation results show that the network shown in Figure 3.8 is indeed unstable under LDQ scheduling. In this simulation all the service rates are 1, and the arrival rate is 0.2. The result is shown in Figure 3.9. As one can see, first all the queues become equal but then do not go to 0 and remain constant in this example.

The basic intuition behind this fact is the following. Consider the simple case of Figure 3.10. We have one queue in each group so no scheduling is needed. However, LDQ serves only one of the queues at a time, and they are virtually in the same group. Consequently, LDQ is not throughput-optimal if we have a cycle in the network.

In the end, we see some simulation results for the Lu-Kumar network under LDQ scheduling. The network topology is acyclic so we expect LDQ to be stable. Simulation results shown in Figure 3.11 verifies this. In this simulation, all the service rates are 1,

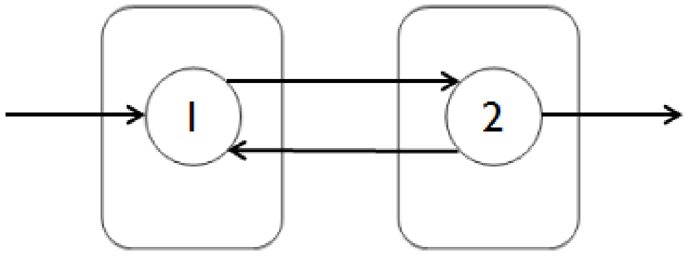

Figure 3.10: A simple cycle

and the arrival rate is 0.4. As one can see, first all the queues become equal and then go to 0 together.

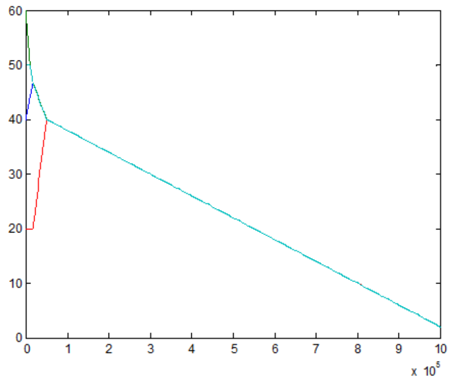

Figure 3.11: LDQ in Lu-Kumar network

Chapter 4

Analysis of Fixed-time Control

In the previous chapters, we addressed the problem of how to design a simple and robust scheduling policy for various types of queueing networks, and analyzed the performance of different policies. In this chapter, we focus on transportation systems, a particular application of queueing networks. We do not try to design a control for this queueing network; instead, we analyze a well-known policy in transportation that is the fixed-time (FT) control. We first introduce and motivate the problem in Section 4.1. In Section 4.2, we consider a single queue, and present the main theoretical results for this simple case. In Section 4.3, we generalize the results for a single queue to the queueing network. Results for the case of periodic demand and FT control are presented in Section 4.4.

4.1 Introduction

Traffic in an urban network is determined by intersection signal control and the pattern of demand. The movement of vehicles is often modeled as a queueing network as, for example, in [56, 52]. Roughly speaking, a vehicle arrives from outside the network at an entry link; travels along a link at a fixed speed; at the end of the link it arrives at an intersection, and joins a queue of vehicles for the next link in its path; the queue is served at a specified service rate when that movement is actuated by the signal; eventually the vehicle leaves the network.

In the U.S. 90 percent of traffic signals follow fixed time (FT) controls, which operate the signal in a fixed periodic cycle, independent of the traffic state. Despite its practical importance, little attention has been paid to understanding how traffic behaves under FT control. Published work has studied queues at a single, isolated intersection, as in [51]. The steady state optimal control of single intersections is studied in [37, 32, 30]. The latter work derives the optimal control settings required to minimize different objectives including queueing delays, but does not address the effect of initial conditions on solution trajectories or their convergence. [29, 30] analyze oversaturated intersections and [67] inroduces an adaptive control for undersaturated networks. But neither work analyzes the behavior of solution trajectories. Signal timing tools used by traffic engineers often employ empirical models [69] in combination with simulations, assuming steady state conditions. But the absence of theory establishing convergence to a unique steady state calls into question whether the traffic flows achieve the performance for which these signals are tuned.

We analyze vehicle movement under two assumptions: first, all the signals have a fixed time (FT) control with the same cycle time or period T; second, vehicles from outside enter the network in periodic streams with the same period. Periodic demands include constant demands, which is the assumption in commercial packages used to design FT controls. Also, if there are intersections with FT controls with different cycles $T_1, \cdots, T_k$, they are all periodic with the same period $T = \mathrm{lcm}\{T_1, \cdots, T_k\}$.

The state of the signalized network at any time t consists of $x(t)$, the vector of all queue lengths, together with the position of all vehicles that are traveling along a link but have not yet reached a queue. We consider a continuous-time system, and vehicles as a fluid instead of as discrete entities. As a result the evolution of the network is described by a delay-differential equation, in which the delay comes from the travel time of a vehicle as it moves from one queue to the next. In an actual transportation network, the arrival and service processes are stochastic. However, an exact analysis of queue-length processes in a stochastic queueing network is very difficult, if not impossible; except for very simple examples such as an isolated intersection, the underlying Markov chain of the system is intractable. Therefore, we consider deterministic arrival and service processes.

Our main contribution is to show that there is a unique periodic trajectory $x^*(t)$ of the queue length vector to which every trajectory $x(t)$ converges; moreover, in case individual vehicles do not circulate in loops, the convergence is in finite time. The periodic orbit determines every possible performance measure, such as delay, travel time, amount of wasted green, and signal progression quality, see [21]. An outstanding open problem is to calculate this periodic orbit without simulation. If this can be done, one would have a computational procedure to design the FT control for a network that optimizes any performance measure.

Our results have some independent mathematical interest. The delay-differential equation is not Lipschitz, and existence and uniqueness of a solution is established using the reflection map of queueing theory [36, 70]. The differential equation is periodic (with period T), and the existence of a periodic orbit is proved using the Poincare map. The global stability of this periodic orbit depends on a monotonicity property reminiscent of that in freeway models [31].

4.2 Single Queue without Routing

Time is continuous, $t \geq 0$. The length or size of a single queue $x(t), t \geq 0$, evolves as

$$\dot{x}(t) = e(t) - b(t), \tag{4.1}$$

with arrivals $e(t) \geq 0$, departures $b(t)$, $t \geq 0$, and initial queue $x(0) = x_0 \geq 0$. Arrivals $e(t)$ are exogenously specified. There is a specified service rate $c(t) \geq 0, t \geq 0$, so departures are given by

$$b(t) = \begin{cases} c(t), & \text{if } x(t) > 0, \\ \in [0, c(t)], & \text{if } x(t) = 0, \\ 0, & \text{if } x(t) < 0. \end{cases} \tag{4.2}$$

Express the departure process as

$$b(t) = c(t) - y(t),\ t \geq 0, \tag{4.3}$$

so $y(t)$ is the rate at which service is unused. From (4.2),

$$y(t) \geq 0, \text{ and } x(t)y(t) = 0,\ \forall t.$$

Rewrite (4.1) as

$$\dot{x}(t) = [e(t) - c(t)] + y(t),$$

or in functional form as

$$x = u + v, \tag{4.4}$$

in which

$$u(t) = x_0 + \int_0^t [e(s) - c(s)]ds \text{ and } v(t) = \int_0^t y(s)ds. \tag{4.5}$$

Then x, v satisfy

$$x(t) \geq 0,\ v(t) \geq 0,\ v(0) = 0,\ y(t) = \dot{v}(t) \geq 0, \text{ and } x(t)\dot{v}(t) = 0,\ \forall t. \tag{4.6}$$

Observe that $v(t)$ has an interpretation as the cumulative unused service (wasted green).

Fact 1 and Theorem 10 are immediate consequences of [36, Theorem 1].

Fact 1. *Suppose x, b satisfy* (4.1)-(4.2). *Define u, v by* (4.3) *and* (4.5). *Then* (4.4) *and* (4.6) *hold. Conversely, suppose u, v, x satisfy* (4.4)-(4.6). *Define $b(t)$ by* (4.3). *Then* (4.1)-(4.2) *hold.*

Proof. Suppose x, b satisfy (4.1)-(4.2) and define u, v by (4.3) and (4.5). Then (4.5) implies (4.4) and (4.2) implies (4.6). Conversely suppose (4.4)-(4.6) hold. Define $b(t)$ by (4.3). Then, by (4.4),

$$\dot{x}(t) = \dot{u}(t) + y(t) = e(t) - c(t) + y(t) = e(t) - b(t),$$

so (4.1) holds. Moreover, by (4.6), $x(t) > 0$ implies $y(t) = c(t) - b(t) = 0$, so $b(t) = c(t)$. If $x(t) = 0$, $t \in (t_1, t_2)$, $\dot{x}(t) = 0, 0 \leq y(t) = c(t) - e(t) \leq c(t)$ and $b(t) = c(t) - y(t) = e(t) \leq c(t)$. This proves (4.2). ■

Theorem 10. *Fix continuous function u with $u(0) = x_0 \geq 0$. There exist unique continuous functions x, v satisfying (4.4)-(4.6). The functions $v = \psi(u)$ and $x = \phi(u)$ are continuous (in sup norm) and given by*

$$v(t) = \sup\{[u(s)]^- \mid 0 \leq s \leq t\}, \text{ with } [z]^- = \max\{-z, 0\}, \tag{4.7}$$

$$x(t) = u(t) + \sup\{[u(s)]^- \mid 0 \leq s \leq t\}. \tag{4.8}$$

Proof. Fix continuous u with $u(0) = x_0 \geq 0$.
Existence Define

$$v(t) = \sup_{0 \leq s \leq t} [u(s)]^- \text{ and } x(t) = u(t) + v(t). \tag{4.9}$$

Then (4.4) holds. $v(0) = [x_0]^- = 0$ and v is increasing. Further, $x(t) \geq 0$, since $x(t) = u(t) + \sup_{0 \leq s \leq t}[u(s)]^- \geq 0$ if $u(t) \geq 0$, and $x(t) \geq u(t) + [-u(t)] \geq 0$ if $u(t) < 0$. Suppose $\dot{v}(t) > 0$. Then $v(t) = \sup_{0 \leq s \leq t}[u(s)]^- = [u(t)]^- = -u(t)$, so $v(t) + u(t) = 0 = x(t)$. This proves (4.6).
Uniqueness Consider any solution

$$q = u + w, \; q \geq 0, \; w \geq 0, \; w(0) = 0, \; \dot{w} \geq 0, \text{ and } q\dot{w} = 0. \tag{4.10}$$

Since $w \geq 0$ and increasing,

$$w(t) = [q(t) - u(t)] = \sup_{0 \leq s \leq t} [q(s) - u(s)] \geq \sup_{0 \leq s \leq t} [u(s)]^- = v(t).$$

If $w(t) > v(t)$, then there is $t_0 < t$ with $w(t_0) > v(t_0)$ and $\dot{w}(t_0) > 0$. But then

$$q(t_0) = w(t_0) + u(t_0) > v(t_0) + u(t_0) = x(t_0) \geq 0.$$

So $q(t_0) > 0$ and $\dot{w}(t_0) > 0$, which contradicts (4.10). So $w(t) = v(t)$ for all t.
Continuity Suppose $v = \psi(u)$ and $v' = \psi(u')$, and $\sup_{0 \leq s \leq t} |u(s) - u'(s)| < \epsilon$. Then

$$\sup_{0 \leq s \leq t} [u(s)]^- \leq \sup_{0 \leq s \leq t} [u'(s)]^- + \epsilon,$$

and so $|v(t) - v'(t)| < \epsilon$. Hence ψ is continuous and so is ϕ. ■

Corollary 1 describes a useful monotonicity property.

Corollary 1. *(Monotonicity) (a) If $u \le u'$ (pointwise), then $v = \psi(u) \ge v' = \psi(u')$. (b) If $x_0 \le x_0'$, $e(t) \le e'(t)$ and $c(t) = c'(t)$, for all t, $x(t) \le x'(t)$ and $b(t) \le b'(t)$, for all t.*

Proof. (a) If $u \le u'$, $[u(s)]^- \ge [u'(s)]^-$, and so $v \ge v'$,
(b) It is enough to show that $x(0) = x'(0)$ implies $x(t) \le x'(t)$ for small t. If $x(0) = x'(0) > 0$ then, from (4.1)-(4.2),

$$\dot{x}(t) = e(t) - c(t) \le e'(t) - c'(t) = \dot{x}'(t),$$

so $x(t) \le x'(t)$ for small t.
If $x(0) = x'(0) = 0$, recall that

$$u(t) = \int_0^t [e(s) - c(s)]ds, \quad u'(t) = \int_0^t [e'(s) - c'(s)]ds.$$

Then $\dot{u}(t) \le \dot{u}'(t)$ and

$$x(t) = u(t) + \sup_{0 \le s \le t} [u(s)]^- = u(t) + [u(s)]^- \text{ for some } s.$$

If $u(s) < 0$,

$$\begin{aligned} x(t) &= \int_0^t \dot{u}(\tau)d\tau - \int_0^s \dot{u}(\tau)d\tau = \int_s^t \dot{u}(\tau)d\tau \\ &\le \int_s^t \dot{u}'(\tau)d\tau = \int_0^t \dot{u}'(\tau)d\tau - \int_0^s \dot{u}'(\tau)d\tau \\ &\le u'(t) + \sup_{0 \le s \le t} [u'(s)]^- = x'(t). \end{aligned}$$

If $u(s) \ge 0$ for $s \le t$, then $x(t) = u(t) \le u'(t) = x'(t)$.
Lastly, if $x'(t) > 0$ then from (4.2) $b'(t) = c'(t) = c(t) \ge b(t)$ and if $x'(t) = x(t) = 0$, $t \in (t_1, t_2)$, $\dot{x}(t) = \dot{x}'(t) = 0$, so from (4.2),

$$b(t) = e(t) \le e'(t) = b'(t).$$

■

Corollary 2. *Suppose $\int_0^t [e(s) - c(s)]ds \to -\infty$ as $t \to \infty$. Consider two solutions x, x' of (4.4) with*

$$x(t) = u(t) + v(t), \quad x'(t) = u'(t) + v'(t).$$

Then there exists T^* such that $x(t) = x'(t)$, $t > T^*$.

Proof. Suppose $u(0) = x_0 < x'_0 = u'(0)$. By Corollary 1, $x(t) \leq x'(t)$ for all t. Suppose $x(t) < x'(t)$ for all t. Then $x'(t) > 0$ for all t and so from (4.6) $v'(t) = 0$ for all t, but then $x'(t) = u'(t) = x'_0 + \int_0^t [e(s) - c(s)]ds \to -\infty$, which contradicts $x'(t) \geq 0$. So there exists T^* such that $x(T^*) = x'(T^*)$, and then $x(t) = x'(t)$ for $t \geq T^*$. ∎

Remark 6. The condition

$$\int_0^t [c(s) - e(s)]ds \to \infty,$$

holds if on average the service rate exceeds the arrival rate by some $\epsilon > 0$. In turn, Corollary 2 says that the effect of initial condition x_0 disappears after a finite time.

Theorem 11. *Suppose $\dot{u}(t)$ is periodic with period T, and $\int_0^T \dot{u}(t)dt < 0$. Then there is a unique periodic trajectory z with period T such that*

$$z(t) = u(t) + v(t).$$

Every solution of

$$x(t) = u'(t) + v'(t), \tag{4.11}$$

coincides with $z(t)$ after some finite time. There exists $t_0 \in [0, T]$ such that $z(t_0) = 0$, i.e. the queue will be cleared in each period.

Proof. Consider the Poincare map $F : x(0) \mapsto x(T)$ and the iterates $x(nT) = F^n(x(0))$. Take $x(0) = 0$. Clearly, $x(T) \geq x(0) = 0$. Hence, by monotonicity,

$$x(0) \leq x(T) \leq x(2T) \leq \cdots .$$

Since $\int_0^T \dot{u}(t)dt < 0$, the queue length is bounded. Since the sequence is bounded and increasing, it converges. The uniqueness and convergence in finite time are immediate results of Corollary 2. Finally, we show that the queue will be cleared in each period. Let z be the unique periodic trajectory. If $z(t) = u(t) + v(t) > 0$ for all t, then $v(t) = 0$ for all t, and since $u(t) \to -\infty$, $z(t) \to -\infty$, but $z(t) \geq 0$. So $z(t_0) = 0$ for some t_0. ∎

Example 5. Consider a single link with constant arrival rate $e(t) = 1$. The cycle time is 1, and the periodic saturation rate is $c(t) = 3$ for $0 \leq t \leq 0.5$, and $c(t) = 0$ for $0.5 < t < 1$. Thus within each cycle the signal is green for time 0.5, and red for time 0.5. During green, 3 vehicles can depart per unit time. Figure 4.1 shows the unique periodic orbit (solid line) starting at $x(0) = 0.5$ and another trajectory (dashed line) starting at $x(0) = 1.5$, which coincides with the periodic orbit after $t = 1.5$. Also shown are the two associated departure processes $b(t)$, which also coincide after $t = 1.5$. Figure 4.2 shows the trajectories of $u(t)$, $v(t)$ and $y(t)$ for the two solutions.

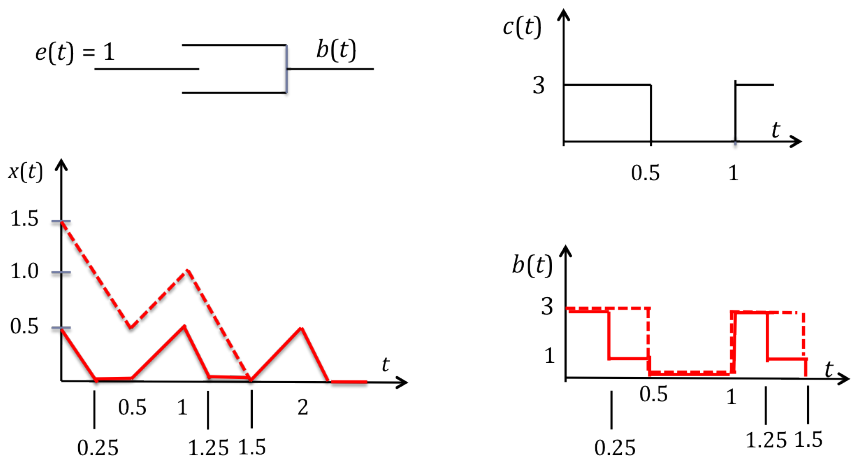

Figure 4.1: A single queue with constant arrival rate, $e(t) = 1$; periodic service rate $c(t)$ with period 1; two solutions converge at $t = 1.5$.

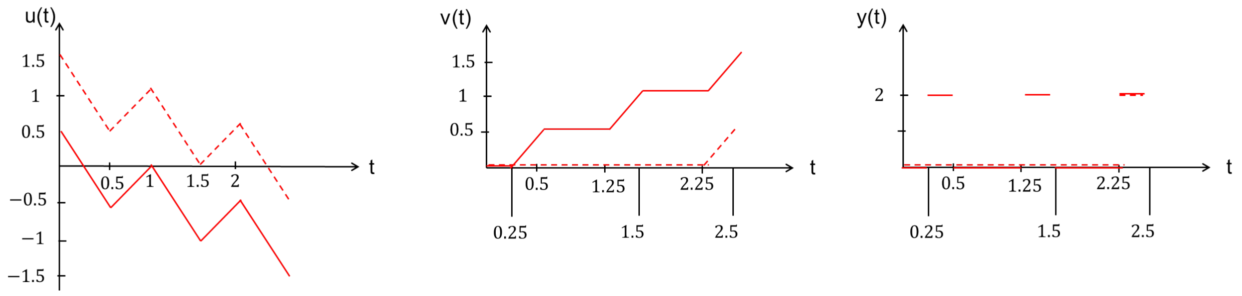

Figure 4.2: Trajectories of $u(t)$, $v(t)$ and $y(t)$ for the two solutions of Example 5.

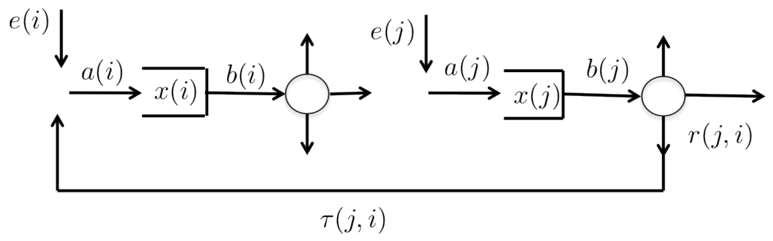

Figure 4.3: Evolution of $x(i)(t)$.

4.3 Network of Queues

Figure 4.3 will help establish notation for the network. A fraction $r(j, i)$ of vehicles leaving queue j will travel along link (j, i) and join queue i after time $\tau(j, i)$, t Vehicles join queue i at rate $a(i)$ either from outside the network at rate $e(i)$ or after being routed from another queue. It is assumed that each queue has infinite storage capacity.

Hence the queueing equations for the network in Figure 4.3 are

$$\begin{aligned}
\dot{x}(i)(t) &= a(i)(t) - b(i)(t),\ t \geq 0, \\
a(i)(t) &= e(i)(t) + \sum_j b(j)(t - \tau(j,i))r(j,i), \\
b(i)(t) &= \begin{cases} c(i)(t), & \text{if } x(i)(t) > 0, \\ \in [0, c(i)(t)], & \text{if } x(i)(t) = 0, \\ 0, & \text{if } x(i)(t) < 0. \end{cases}
\end{aligned}$$

Above $c(i)(t)$ is the saturation flow or service rate at which queue i is served.

Express the departure process as

$$b(i)(t) = c(i)(t) - y(i)(t), t \geq 0. \tag{4.12}$$

Then $y(i)(t) \in [0, c(i)(t)]$ is the unused service rate,

$$y(i)(t) \geq 0, \text{ and } x(i)(t)y(i)(t) \equiv 0. \tag{4.13}$$

Rewrite the system equations as

$$\dot{x}(i)(t) = e(i)(t) - c(i)(t) + \sum_j b(j)(t - \tau(j,i))r(j,i) + y(i)(t), \tag{4.14}$$

or in functional form as

$$x(i) = u(i) + v(i), \tag{4.15}$$

in which

$$\begin{aligned}
u(i)(t) &= x(i)(0) + \int_0^t \{e(i)(s) - c(i)(s) + \sum_j b(j)(s - \tau(j,i))r(j,i)\}ds, && (4.16) \\
b(j)(s) &= c(j)(s) - y(j)(s) && (4.17) \\
v(i)(t) &= \int_0^t y(i)(s)ds. && (4.18)
\end{aligned}$$

Fix the external arrivals and saturation flows, $\{e(i)(t), c(i)(t), t \geq 0\}$. Suppose $0 < \tau = \min\{\tau(i,j)\} \leq \bar{\tau} = \max\{\tau(i,j)\}$. Fix initial conditions

$$x(i)(0) \geq 0 \text{ and } \{b(i)(s) \geq 0, s \in [-\bar{\tau}, 0]\}. \tag{4.19}$$

This determines $u(i)(s), s \in [0, \tau]$. By Theorem 10, there exist unique $\{x(i)(t), v(i)(t)\}$ satisfying (4.13)-(4.14) for $t \in [0, \tau]$. In turn this fixes new initial conditions at time τ similar to (4.19):

$$\{x(i)(\tau) \geq 0\} \text{ and } \{b(i)(s) = c(i)(s) - y(i)(s) \geq 0, s \in [-\bar{\tau} + \tau, \tau]\},$$

so that again by Theorem 10 the solution can be extended to $[\tau, 2\tau]$. Proceeding step wise in this way leads to the next result.

Theorem 12. *Fix arrivals $\{e(i)(t)\}$, saturation rates $\{c(i)(t)\}$, and routing ratios $\{r(i,j)\}$. Fix initial conditions $\{x(i)(0) \geq 0\}, \{b(i)(s), s \in [-\bar{\tau}, 0]\}$. Then there are unique functions $\{x(i)(t), b(i)(t), v(i)(t),\ t \geq 0\}$ satisfying* (4.12)-(4.14).

Corollary 3. *(Monotonicity) Suppose $x(i)(0) \leq x'(i)(0), 0 \leq b(i)(s) \leq b'(i)(s), s \in [-\bar{\tau}, 0]$, $e(i)(t) \leq e'(i)(t)$, $c(i)(t) = c'(i)(t)$, $t \geq 0$. Then $x(i)(t) \leq x'(i)(t)$, $b(i)(t) \leq b'(i)(t)$, $u(i)(t) \leq u'(i)(t)$, $v(i)(t) \geq v'(i)(t)$, all t.*

Proof. The result follows by applying Corollary 1 successively over $[0, \tau], [\tau, 2\tau], \cdots$ ■

Corollary 4. *With the same notation and hypothesis as in Corollary 3, suppose $e(i)(t) = e'(i)(t)$, and define $z(i)(t) = x'(i)(t) - x(i)(t)$. Then*

$$\begin{aligned}
\dot{x}(i)(t) &= e(i)(t) - b(i)(t) + \sum b(j)(t - \tau(j,i))r(j,i), \\
\dot{x}'(i)(t) &= e'(i)(t) - b'(i)(t) + \sum b'(j)(t - \tau(j,i))r(j,i), \\
\dot{z}(i)(t) &= -[b'(i)(t) - b(i)(t)] + \sum [b'(j)(t - \tau(j,i)) - b(j)(t - \tau(j,i))]r(j,i).
\end{aligned}$$

Then $z(i)(t) \geq 0$ for all t.

4.4 Periodic Solution

We now consider FT control. Suppose that the external arrivals $e(i)(t)$ and saturation flow rates $c(i)(t)$ are all periodic with the same period T. Let

$$\bar{e}(i) = \frac{1}{T}\int_0^T e(i)(t)dt,\ \bar{c}(i) = \frac{1}{T}\int_0^T c(i)(t)dt. \tag{4.20}$$

We establish a necessary condition for the existence of a periodic solution to (4.12)-(4.14) with period T. Let

$$\bar{b}(i) = \frac{1}{T}\int_0^T b(i)(t)dt = \frac{1}{T}\int_0^T b(i)(t - \tau(i,j))dt,\ \bar{y}(i) = \frac{1}{T}\int_0^T y(i)(t)dt. \tag{4.21}$$

Using this notation in (4.14), and integrating over $[0, T]$ for a periodic solution x gives

$$\begin{aligned}
0 &= x(i)(T) - x(i)(0) = \bar{e}(i) - \bar{c}(i) + \sum_j \bar{b}(j)r(j,i) + \bar{y}(i) \\
&= \bar{e}(i) - \bar{c}(i) + \sum_j \bar{c}(j)r(j,i) + \bar{y}(i) - \sum_j \bar{y}(j)r(j,i),
\end{aligned}$$

or, in vector form, denoting the routing matrix $R = \{r(i,j)\}$,

$$0 = \bar{e} - [I - R^T]\bar{c} + [I - R^T]\bar{y}.$$

Since every vehicle eventually leaves, $[I - R^T]^{-1} = I + R^T + R^{2T} + \cdots \geq 0$ exists and so the preceding condition becomes

$$0 = [I - R^T]^{-1}\bar{e} - \bar{c} + \bar{y}, \tag{4.22}$$

so, for a periodic solution to exist, one must have

$$\bar{c} = [I - R^T]^{-1}\bar{e} + \bar{y} \geq [I - R^T]^{-1}\bar{e}.$$

We impose the slightly stronger *stability condition*: there exists $\epsilon > 0$ such that

$$\bar{c} > [I - R^T]^{-1}\bar{e} + \epsilon\mathbf{1}, \tag{4.23}$$

which says that on average the service rate for each queue exceeds the total arrival rate.

Remark 7. In the non-periodic case the stability condition (4.23) may be replaced by

$$\int_0^{NT} c(t)dt > [I - R^T]^{-1}\int_0^{NT} e(t)dt + \epsilon NT\mathbf{1}, \text{ for all } N, \text{ for some } \epsilon > 0. \tag{4.24}$$

Lemmas 18 and 19 hold under this stability condition.

Lemma 18. *Every trajectory $x(t)$ of vehicle queue lengths is bounded.*

Proof. Let x, u, v, y be a solution of (4.15)-(4.18). Then

$$\dot{x}(t) = e(t) - c(t) + R^T b(t) + y(t) + \delta(t), \tag{4.25}$$

in which

$$\delta(i)(t) = \sum_j [b(j)(t - \tau(j,i)) - b(j)(t)]r(j,i).$$

So

$$\int_s^t \delta(i)(r)dr = \sum_j r(j,i)\big[\int_{s-\tau(j,i)}^{s} b(j)(s) - \int_{t-\tau(j,i)}^{t} b(j)(s)\big]ds.$$

Since $\tau(j,i) \leq \bar{\tau}$, and $b(j)(s) \leq c(j)(s)$ is bounded, it follows that $|\int_s^t \delta(r)dr|$ is bounded for all s, t. So $|\int_s^t \delta(r)dr| \leq d\mathbf{1}$ for some constant d.

We show that if $x(i)(t_0) > NT\bar{c}(i)$, then $x(i)(t_0 + NT) - x(i)(t_0) < 0$ for some constant $N > \frac{d}{T\epsilon}$. This is sufficient to show that $x(i)(t)$ is bounded, since the queue-length change per period is bounded. Suppose that $x(i)(t_0) > NT\bar{c}(i)$. Since $x(i)(t + T) - x(i)(t) \geq -T\bar{c}(i)$, so $x(i)(t) > 0$ for $t_0 \leq t \leq t_0 + NT$ and $y(i)(t) = 0$ for $t_0 \leq t \leq t_0 + NT$. Integrating both sides of (4.25) gives

$$\begin{aligned} x(i)(t + NT) - x(i)(t) &\leq NT\bar{e}(i) + NT\sum_j r(j,i)\bar{c}(j) - NT\bar{c}(i) + d \quad (4.26)\\ &\leq -NT\epsilon + d < 0, \quad (4.27)\end{aligned}$$

in which (4.26) uses the fact that $\bar{b}(j) \leq \bar{c}(j)$ and (4.27) follows from the stability condition and the inequality $N > \frac{d}{T\epsilon}$. ■

4.4.1 Effect of Initial Conditions

Suppose the stability condition (4.23) holds. By Lemma 18 the queue lengths are bounded, so from (4.26)

$$\int_0^t y(r)dr \to \infty,$$

component wise. Thus the cumulative unused service at every queue is unbounded. This implies that, independent of the service discipline (whether first in first out or something else), so long as the discipline is work conserving (i.e., a queue is served if it is non-empty, see (4.2)), every vehicle in the initial condition will eventually leave the network.

The *state* of the network at time t is the pair $(x(t), \beta(t))$ where $\beta(t)$ is the history of departures over time $[t-\bar{\tau}, t]$, i.e., $\beta(t)$ is the function: $s \in [t-\bar{\tau}, t] \mapsto b(s)$. We want to show that trajectories starting from two different initial conditions, say $(x(0), \beta(0))$ and $(x'(0), \beta'(0))$, will eventually converge. Because of the monotonicity property, Corollary 3, we may take one of the initial conditions to be zero.

Lemma 19. *Let $x(t), \beta(t)$ be the trajectory starting from $(x(0), \beta(0))$, and let $z(t), \beta'(t)$ be the trajectory starting from $(0,0)$. Then the two trajectories converge:*

$$\lim_{t\to\infty} |x(t) - z(t)| \to 0, \text{ and } \lim_{t\to\infty} \sup_{s\in[t-\bar{\tau},t]} [\int_0^s [b(r) - b'(r)]dr \to 0, \tag{4.28}$$

Proof. The proof relies on an intuitive argument. Consider the trajectory $(x(t), \beta(t))$. Color the vehicles in the initial state $(x(0), \beta(0))$ red, and color all vehicles entering the network after time 0, black. In each queue there will be black and red vehicles. Change the service discipline in each queue so that all black vehicles are served ahead of every red vehicle. Then the red vehicles do not interfere with the movement of black vehicles and so the number of black vehicles in the queues and along the links will be identical to $(z(t), \beta'(t))$. On the other hand the total number (red plus black) vehicles in the queues and along the links will be identical to $(x(t), \beta(t))$. Because of the stability condition, every vehicle in the initial queue $x(0)$ will eventually leave the network, that is $x(t) - z(t) \to 0$, as $t \to \infty$. But then the second part of (4.28) follows. ■

Remark 8. Suppose the arrivals and service processes, e and c, are stochastic and the stability condition (4.24) holds almost surely. Then the effect of the initial state on the queue length process will disappear over time since (4.28) will hold almost surely along every sample path.

4.4.2 Existence of Periodic Solutions

We prove the existence of a unique periodic solution to which all trajectories converge.

Theorem 13. *There exists a unique periodic state trajectory (x^*, β^*), with period T, to which every trajectory converges.*

Proof. Consider the Poincare map $F : (x(0), \beta(0)) \mapsto (x(T), \beta(T))$ and the iterates $(x(nT), \beta(nT)) = F(x((n-1)T, \beta((n-1)T))$. Take $x(0) = 0, \beta(0) = 0$. Then by monotonicity $(x(T) \geq x(0) = 0, \beta(T) \geq \beta(0) = 0)$, and hence by repeatedly using monotonicity we get:

$$x(0) \leq x(T) \leq x(2T) \leq \cdots, \quad \beta(0) \leq \beta(T) \leq \beta(2T) \leq \cdots.$$

Thus this sequence of states is increasing. By Lemma 18 the sequence is bounded, so it converges to the state say (x^*, β^*). By Theorem 10, F is continuous, so $F((x^*, \beta^*)) = (x^*, \beta^*)$, and the trajectory from this state is periodic with period T. ■

Corollary 5. *In the periodic trajectory x^* every queue clears in each period, i.e., for each i there exists $t_i \in [0, T]$ such that $x(i)(t_i) = 0$. In each period, the cumulative unused service is $\bar{y} = \bar{c} - [I - R^T]^{-1}\bar{e}$.*

Proof. If $x(i)(t) > 0$ for $t \in [0, T]$, $y(i)(t) = 0$ for $t \in [0, T]$ and so $\bar{y}(i) = 0$, which contradicts the stability condition (4.22), (4.23). ■

4.4.3 Finite Time Convergence

Theorem 14. *Suppose every vehicle leaves after visiting at most K queues. Then every trajectory converges to the periodic trajectory in finite time.*

Proof. Revisit the proof of Lemma 19. Consider the trajectory $(x(t), \beta(t))$ starting in the initial state $(x(0), \beta(0))$. Color the vehicles in the initial state red, and the vehicles arriving after time 0, black. Color all vehicles in the state starting in state $(0, 0)$, black. The red vehicles will potentially be served infinitely often in each queue and so they will all be gone after a finite time. At that time the two trajectories wil coincide and will agree with the periodic trajectory. ■

Example 6. If vehicles can circulate indefinitely, convergence may take infinite time. Figure 4.4 shows a network with a single queue with initial size $x(0) = x_0$ and periodic service rate $c(t)$ with period 1, $c(t) = 1$ for $0 \leq t \leq 1/2$ and $c(t) = 0$ for $1/2 < t \leq 1$. One-half $(r = 1/2)$ of the departing vehicles return for service after travel time $\tau = 1/2$; the other vehicles leave. Suppose $x_0 < 1/2$. Then all vehicles will depart by time x_0, one-half of them will leave and one-half or $x_0/2$ will re-enter the queue during time $[1/2, 1/2 + x_0]$. Since the service rate is 0 until time 1, so $x(t) = x_0/2$, $1/2 + x_0 < t < 1$. At time 1, the queue is $x_0/2$ and there is no vehicle traveling in the link. By induction, we have $x(n) = (1/2)^n x_0$, so convergence takes infinite time.

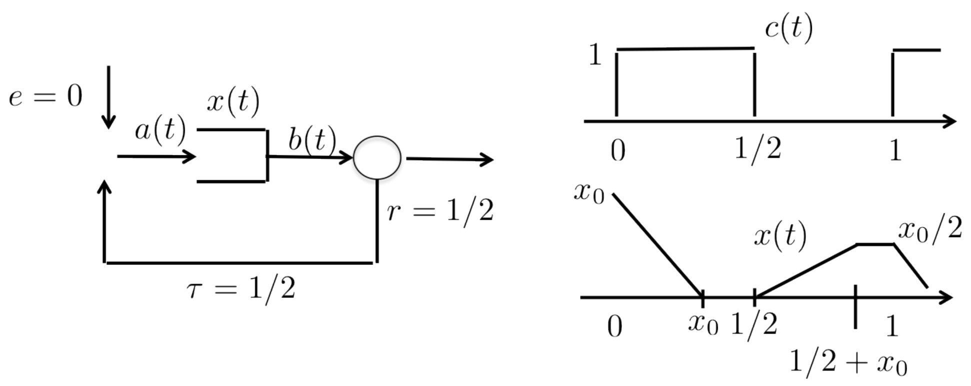

Figure 4.4: Vehicles recirculate (left); periodic service rate (right, top); queue $x(t)$ (right, bottom).

Example 7. From the periodic trajectory one can calculate performance measures such as average delay. In Figure 4.5 the departure process $b(t)$ of Figure 3.2 is the arrival process at the next intersection with the service rate $c_1(t)$ and queue $x_1(t) \equiv 0$ or service rate $c_2(t)$ and queue $x_2(t)$. The average delay per vehicle at the second

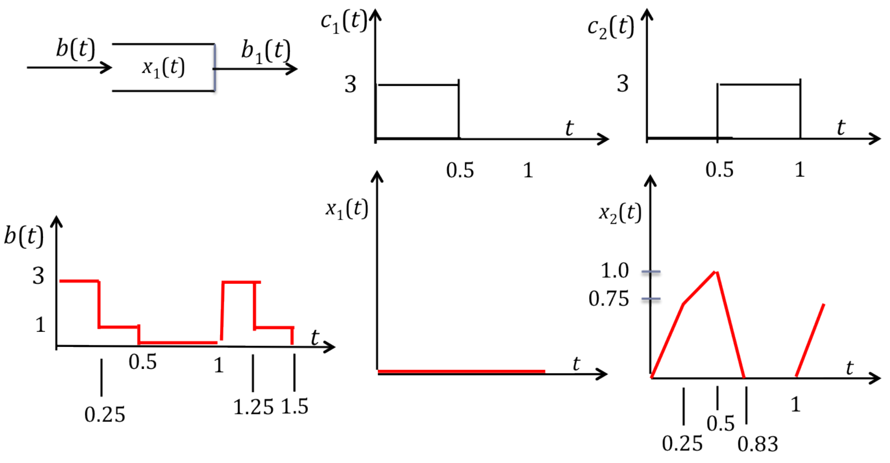

Figure 4.5: Departures $b(t)$ of Figure 3.2 enter an intersection with service rate $c_1(t)$ or $c_2(t)$.

intersection therefore is

$$\int_0^1 x_1(t)dt = 0, \text{ or } \int_0^1 x_2(t) = \frac{23}{48} \approx 0.48.$$

Hence, depending on the offset of the second signal, the average delay at the second intersection can take any value in [0, 0.48]. The average delay in the first intersection with constant arrival can be calculated from the plot of $x(t)$ in Figure 3.2 as

$$\int_0^1 x(t)dt = \frac{3}{16} \approx 0.19.$$

4.4.4 Discussion on Finite Storage Capacity

In the discussion so far it has been assumed that every queue has infinite storage capacity so service is never blocked. We now modify this assumption. Suppose queue i has storage capacity $\xi(i)$. This means that if $x(i)(t)$ reaches $\xi(i)$, additional vehicles cannot be accommodated and service to queues upstream of queue i is blocked. More precisely, the service rate for queue i is changed from $c(i)(t)$ to $c(i)(t)s(i)(t)$ in which

$$s(i)(t) = \begin{cases} 1, & \text{if } x(j)(t) < \xi(j) \text{ for all } j \text{ such that } r(i,j) > 0, \\ 0, & \text{if } x(j)(t) = \xi(j) \text{ for some } j \text{ such that } r(i,j) > 0. \end{cases} \tag{4.29}$$

The system equations (4.14)-(4.18) remain the same except that $c(i)(t)$ is everywhere replaced by $c(i)(t) \times s(i)(t)$. The formulation (4.29) implies that service from i to k is blocked even if $x(k)(t) < \xi(k)$. This is a kind of first in first out assumption. We can avoid this by positing a separate queue for each movement as in [67]. The example below is unaffected in either case.

The possibility of blocking can destroy the previous results. We can see this by examining again the single queue system of Figure 4.4. Suppose that the queue has a storage capacity $\xi = 1$. In this case the effective service rate is $c(t) \times s(t)$ and $s(t) = \mathbf{1}(x(t) < 1)$. Suppose $e(t) = 0$ for all t, and $x(t_0) = 1$. Then $x(t) = 1$ and $b(t) = 0$, for $t \geq t_0$ for all t, and the system is in gridlock. If $x(t_0) = 0$, then $x(t) = 0$, for $t \geq t_0$ is another solution.

Suppose there is a constant arrival $e(t) = \bar{e} < 1/4$. If $\xi = \infty$, the stability condition (4.23) holds and there will be a periodic solution, x^*. Suppose $\max_t x^*(t) = \bar{\xi}$. If the storage capacity $\xi > \bar{\xi}$, then x^* is also a solution.

Above, the service rate $c(i)(t) \times s(i)(t)$ is state-dependent and so the results above do *not* apply. In actuated traffic control, as opposed to FT control, the service rate indeed depends on the traffic state, so studying (4.14) for state-dependent service, $c(i, x(t), t)$, is important. The fundamental results on existence and monotonicity in case that $c(i, x, t)$ is Lipschitz in x are obtained in [62].

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we considered the problem of how to design an efficient, simple and robust scheduling policy for various queueing networks. We also analyzed the performance of two simple well-known scheduling policies: longest-queue scheduling policy for multiclass queueing networks and fixed-time control policy for transportation networks.

We proposed a new fork-join queueing network for scheduling jobs that are represented as directed acyclic graphs to abstractly model job scheduling in data centers and other complicated processing networks. We further developed a novel methodology for designing robust scheduling policies for these queueing networks. The key idea of our design is to use the queue-length changes information to learn the correct server's capacity allocation by stochastic gradient projection method. Our scheduling policy is oblivious to the knowledge of arrival rates and service rates of tasks in the network.

Longest-queue scheduling policy is a robust and local scheduling policy that is of great practical interest for its simplicity. In this policy, each server works on the longest queue that the server can serve at each time. Unfortunately, the throughput-optimality of longest-queue scheduling is still open for multiclass queueing network, which is a well-known and well-studied network model in queueing theory. In this thesis, we tackled this open problem and resolved it for a special case that the network has two servers, and each server can serve two queues.

Finally, we focused on transportation network as an important application of queueing networks. We modeled a network of signalized intersections as a queueing network. In most cases, these signals are controlled by fixed-time policies. We analyzed this queueing network under the fixed-time scheduling policy, and showed that every solution of the system converges to a unique periodic solution, independent of the initial condition.

5.2 Future Research Directions

There are many interesting future research directions of this work. In the remainder of this chapter, we categorize them into two topics and provide a brief overview of them.

5.2.1 Designing Robust Scheduling Policies

In Chapter 2, we designed a robust policy for the case that μ_{kj}, the service rate of task k served by server j, can be factorized to two terms: $\mu_{kj} = \mu_k \alpha_j$. One future research direction is to find a robust scheduling policy for generic service rates, which cannot be necessary factorized to a task-dependent rate and a server-dependent rate.

Further, in Chapter 2, we designed a policy that required the knowledge of the queue-lengths in the network at the current time slot, and at the previous time slot. A future direction is to investigate whether there exists a throughput-optimal policy which is only dependent on the queue-size information in the network in the current state.

In the case of flexible queueing networks, we proposed a scheduling policy that is robust to arrival and service rates, but it requires the knowledge of routing probabilities in the network. An interesting future research direction is to design a scheduling policy that is oblivious to the routing structure of the network.

A natural robust scheduling policy is to serve the longest queue per server. This policy is fully local and oblivious to all network parameters. Even for simple network models such as multiclass queueing networks, whether longest-queue scheduling is throughput-optimal or not is still an open problem. Resolving this open problem can be of great intellectual and practical interest.

5.2.2 Transportation Networks

In Chapter 4, we considered a transportation network and showed that under fixed-time control there is a unique periodic trajectory, which is globally asymptotically stable, that is, every trajectory converges to this periodic trajectory. From the periodic trajectory one can easily calculate every possible performance measure such as delay, travel time, amount of time service is wasted, and progression quality. Thus, an important question for future research is to find an algorithm to calculate the periodic orbit without simulation. Another question is to study the behavior of traffic in networks in which the service rate is state-dependent. [67] has an interesting conjecture that if the control is a function of the queue-lengths, there will again be a unique asymptotically stable trajectory.

There are many other possible future research directions in transportation networks. A major difficulty in designing efficient signal control policies in a network of intersections is the switching cost, that is an all-red signal for the intersection due to safety reasons. The switching cost significantly complicates the design of efficient controls not only for a network of intersections, but also for a single intersection. One future direction is to investigate the effects of switching cost on different scheduling policies, and design a simple efficient signal control in the presence of switching cost. Another technical difficulty that rises in transportation networks is the existence of non-negligible delay on the links of the network. Exploring how these link delays can affect well-known throughput-optimal policies such as Max-pressure policy can be of great interest.

Bibliography

[1] K Aboudolas, M Papageorgiou, and E Kosmatopoulos. “Store-and-forward based methods for the signal control problem in large-scale congested urban road networks”. In: *Transportation Research Part C: Emerging Technologies* 17.2 (2009), pp. 163–174.

[2] K Aboudolas et al. “A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks”. In: *Transportation Research Part C: Emerging Technologies* 18.5 (2010), pp. 680–694.

[3] Sigrún Andradóttir, Hayriye Ayhan, and Douglas G Down. “Dynamic server allocation for queueing networks with flexible servers”. In: *Operations Research* 51.6 (2003), pp. 952–968.

[4] Rami Atar, Avishai Mandelbaum, and Asaf Zviran. “Control of Fork-Join Networks in heavy traffic”. In: *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE. 2012, pp. 823–830.

[5] Francois Baccelli and Zhen Liu. “On the execution of parallel programs on multiprocessor systems—a queuing theory approach”. In: *Journal of the ACM (JACM)* 37.2 (1990), pp. 373–414.

[6] Francois Baccelli and A Makowski. “Simple computable bounds for the fork-join queue”. In: *John Hopkins Conf. Information Science*. 1985.

[7] Francois Baccelli, Armand M Makowski, and Adam Shwartz. “The fork-join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds”. In: *Advances in Applied Probability* (1989), pp. 629–660.

[8] Francois Baccelli, William A Massey, and Don Towsley. “Acyclic fork-join queuing networks”. In: *Journal of the ACM (JACM)* 36.3 (1989), pp. 615–642.

[9] Golshid Baharian and Tolga Tezcan. “Stability analysis of parallel server systems under longest queue first”. In: *Mathematical Methods of Operations Research* 74.2 (2011), pp. 257–279.

[10] Nicholas Bambos and Jean Walrand. “On stability and performance of parallel processing systems”. In: *Journal of the ACM (JACM)* 38.2 (1991), pp. 429–452.

[11] Vivek S Borkar. “Stochastic approximation”. In: *Cambridge Books* (2008).

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[13] Maury Bramson. “Convergence to equilibria for fluid models of FIFO queueing networks”. In: *Queueing Systems* 22.1-2 (1996), pp. 5–45.

[14] Maury Bramson. “Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks”. In: *Queueing Systems* 23.1-4 (1996), pp. 1–26.

[15] Maury Bramson. *Stability of queueing networks.* Springer, 2008.

[16] Jim G Dai. “On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models”. In: *The Annals of Applied Probability* (1995), pp. 49–77.

[17] Jim G Dai. *Stability of fluid and stochastic processing networks.* University of Aarhus. Centre for Mathematical Physics and Stochastics (MaPhySto)[MPS], 1999.

[18] Jim G Dai and Wuqin Lin. “Maximum pressure policies in stochastic processing networks”. In: *Operations Research* 53.2 (2005), pp. 197–218.

[19] Jim G Dai and Sean P Meyn. “Stability and convergence of moments for multiclass queueing networks via fluid limit models”. In: *Automatic Control, IEEE Transactions on* 40.11 (1995), pp. 1889–1904.

[20] Jim G Dai and Gideon Weiss. “Stability and instability of fluid models for reentrant lines”. In: *Mathematics of Operations Research* 21.1 (1996), pp. 115–134.

[21] Christopher M Day et al. *Performance measures for traffic signal systems: An outcome-oriented approach.* Tech. rep. 2014.

[22] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[23] Luca Dieci and Luciano Lopez. “Sliding motion in Filippov differential systems: theoretical results and a computational approach”. In: *SIAM Journal on Numerical Analysis* 47.3 (2009), pp. 2023–2051.

[24] Antonis Dimakis and Jean Walrand. “Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits”. In: *Advances in Applied probability* (2006), pp. 505–521.

[25] Rick Durrett. *Probability: theory and examples.* Cambridge university press, 2010.

[26] Alekseui F Filippov and Felix M Arscott. *Differential equations with discontinuous righthand sides: control systems.* Vol. 18. Springer Science & Business Media, 1988.

[27] Leopold Flatto. “Two parallel queues created by arrivals with two demands II”. In: *SIAM Journal on Applied Mathematics* 45.5 (1985), pp. 861–878.

[28] Leopold Flatto and S Hahn. “Two parallel queues created by arrivals with two demands I”. In: *SIAM Journal on Applied Mathematics* 44.5 (1984), pp. 1041–1053.

[29] Denos C Gazis. “Optimum control of a system of oversaturated intersections”. In: *Operations Research* 12.6 (1964), pp. 815–831.

[30] Denos C Gazis. *Traffic theory*. Vol. 50. Springer Science & Business Media, 2002.

[31] Gabriel Gomes et al. “Behavior of the cell transmission model and effectiveness of ramp metering”. In: *Transportation Research Part C: Emerging Technologies* 16.4 (2008), pp. 485–513.

[32] Jack Haddad et al. “Constrained optimal steady-state control for isolated traffic intersections”. In: *Control Theory and Technology* 12.1 (2014), pp. 84–94.

[33] J Michael Harrison. “Brownian models of open processing networks: Canonical representation of workload”. In: *Annals of Applied Probability* (2000), pp. 75–103.

[34] J Michael Harrison. *Brownian motion and stochastic flow systems*. Wiley New York, 1985.

[35] J Michael Harrison and Vien Nguyen. “Brownian models of multiclass queueing networks: Current status and open problems”. In: *Queueing Systems* 13.1-3 (1993), pp. 5–40.

[36] J Michael Harrison and Martin I Reiman. “Reflected Brownian motion on an orthant”. In: *The Annals of Probability* (1981), pp. 302–308.

[37] G Improta and GE Cantarella. “Control system design for an individual signalized junction”. In: *Transportation Research Part B: Methodological* 18.2 (1984), pp. 147–167.

[38] Michael Isard et al. “Dryad: distributed data-parallel programs from sequential building blocks”. In: *ACM SIGOPS Operating Systems Review*. Vol. 41. 3. ACM. 2007, pp. 59–72.

[39] Libin Jiang and Jean Walrand. “A distributed CSMA algorithm for throughput and utility maximization in wireless networks”. In: *IEEE/ACM Transactions on Networking (TON)* 18.3 (2010), pp. 960–972.

[40] Changhee Joo, Xiaojun Lin, and Ness B Shroff. “Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks”. In: *IEEE/ACM Transactions on Networking (TON)* 17.4 (2009), pp. 1132–1145.

[41] Srikanth Kandula et al. “The nature of data center traffic: measurements & analysis”. In: *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM. 2009, pp. 202–208.

[42] Frank P Kelly. “Networks of queues”. In: *Advances in Applied Probability* (1976), pp. 416–432.

[43] Panagiotis Konstantopoulos and Jean Walrand. “Stationary and stability of fork-join networks”. In: *Journal of Applied Probability* (1989), pp. 604–614.

[44] PR Kumar and Sean P Meyn. “Stability of queueing networks and scheduling policies”. In: *Automatic Control, IEEE Transactions on* 40.2 (1995), pp. 251–260.

[45] Sunil Kumar and PR Kumar. “Fluctuation smoothing policies are stable for stochastic re-entrant lines”. In: *Discrete Event Dynamic Systems* 6.4 (1996), pp. 361–370.

[46] Steve H Lu and PR Kumar. “Distributed scheduling based on due dates and buffer priorities”. In: *Automatic Control, IEEE Transactions on* 36.12 (1991), pp. 1406–1416.

[47] Vadim A Malyshev. “Networks and dynamical systems”. In: *Advances in applied probability* (1993), pp. 140–175.

[48] Avishai Mandelbaum and Alexander L Stolyar. “Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized cμ-rule”. In: *Operations Research* 52.6 (2004), pp. 836–855.

[49] M Mandelbaum and B Avi-Itzhak. “Introduction to queueing with splitting and matching”. In: *Israel Journal of Technology.* Vol. 6. 1968.

[50] Nicholas W McKeown. “Scheduling algorithms for input-queued cell switches”. PhD thesis. UC Berkeley, 1995.

[51] Alan J Miller. “Settings for fixed-cycle traffic signals”. In: *OR* (1963), pp. 373–386.

[52] Pitu Mirchandani and Larry Head. “A real-time traffic signal control system: architecture, algorithms, and analysis”. In: *Transportation Research Part C: Emerging Technologies* 9.6 (2001), pp. 415–432.

[53] Ajith Muralidharan, Ramtin Pedarsani, and Pravin Varaiya. “Analysis of fixed-time control”. In: *Transportation Research Part B: Methodological* 73 (2015), pp. 81–90.

[54] Viên Nguyen. “Processing networks with parallel and sequential tasks: Heavy traffic analysis and Brownian limits”. In: *The Annals of Applied Probability* (1993), pp. 28–55.

[55] Viên Nguyen. “The trouble with diversity: Fork-join networks with heterogeneous customer population”. In: *The Annals of Applied Probability* (1994), pp. 1–25.

[56] Markos Papageorgiou et al. “Review of road traffic control strategies”. In: *Proceedings of the IEEE* 91.12 (2003), pp. 2043–2067.

[57] Ramtin Pedarsani and Jean Walrand. "Stability of multiclass queueing networks under longest-queue and longest-dominating-queue scheduling". In: *to appear in Journal of Applied Probability* (2016).

[58] Ramtin Pedarsani, Jean Walrand, and Yuan Zhong. "Robust scheduling and congestion control for flexible queueing networks". In: *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE. 2014, pp. 467–471.

[59] Ramtin Pedarsani, Jean Walrand, and Yuan Zhong. "Robust scheduling in a flexible fork-join network". In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 3669–3676.

[60] Ramtin Pedarsani, Jean Walrand, and Yuan Zhong. "Scheduling tasks with precedence constraints on multiple servers". In: *Allerton Conference on Communication, Control, and Computing*. 2014.

[61] Robert J Plemmons. "M-matrix characterizations. I—nonsingular M-matrices". In: *Linear Algebra and its Applications* 18.2 (1977), pp. 175–188.

[62] S Ramasubramanian. "A subsidy-surplus model and the Skorokhod problem in an orthant". In: *Mathematics of Operations Research* 25.3 (2000), pp. 509–538.

[63] Alexander L Stolyar. "On the stability of multiclass queueing networks: A relaxed sufficient condition via limiting fluid processes". In: *Markov Processes and Related Fields* 1.4 (1995), pp. 491–512.

[64] Alexander L Stolyar and Elena Yudovina. "Tightness of invariant distributions of a large-scale flexible service system under a priority discipline". In: *Stochastic Systems* 2.2 (2012), pp. 381–408.

[65] Leandros Tassiulas and Anthony Ephremides. "Dynamic server allocation to parallel queues with randomly varying connectivity". In: *Information Theory, IEEE Transactions on* 39.2 (1993), pp. 466–478.

[66] Leandros Tassiulas and Anthony Ephremides. "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks". In: *Automatic Control, IEEE Transactions on* 37.12 (1992), pp. 1936–1948.

[67] Pravin Varaiya. "Max pressure control of a network of signalized intersections". In: *Transportation Research Part C: Emerging Technologies* 36 (2013), pp. 177–195.

[68] Subir Varma. "Heavy and light traffic approximations for queues with synchronization constraints". In: *Ph.D. thesis, Department of Electrical Engineering, University of Maryland* (1990).

[69] Fo Vo Webster. *Traffic signal settings*. Tech. rep. 1958.

[70] Ward Whitt. "The reflection map with discontinuities". In: *Mathematics of Operations Research* 26.3 (2001), pp. 447–484.

[71] Matei Zaharia et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association. 2012, pp. 2–2.