

Copyright  
by  
Shreeshankar Ravishankar Bodas  
2010

The Dissertation Committee for Shreeshankar Ravishankar Bodas certifies that this is the approved version of the following dissertation:

## High-performance Scheduling Algorithms for Wireless Networks

Committee:

---

Sanjay Shakkottai, Supervisor

---

Sriram Vishwanath, Supervisor

---

Constantine Caramanis

---

Gustavo de Veciana

---

John Hasenbein

---

R. Srikant

**High-performance Scheduling Algorithms for Wireless  
Networks**

by

**Shreeshankar Ravishankar Bodas, B.Tech., M.S.E.**

**DISSERTATION**

Presented to the Faculty of the Graduate School of  
The University of Texas at Austin  
in Partial Fulfillment  
of the Requirements  
for the Degree of

**DOCTOR OF PHILOSOPHY**

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2010

To Aai, Baba, Surashree, and the memory of my grandfather

## Acknowledgments

I consider myself extremely fortunate to have had two wonderful supervisors, Sanjay Shakkottai and Sriram Vishwanath, and I would like to take this opportunity to thank them for their guidance, support and encouragement throughout my years as a PhD student at UT Austin. I also thank the NSF for funding my research.

Sanjay has been a fantastic mentor and a person I have looked up to as a role-model. The depth of his technical knowledge and his infinite energy for exploring new stuff continue to impress me. He has helped me in numerous ways throughout my PhD: his help with formulating a meaningful research problem, his patience when listening to my analysis of the problem, his constructive criticism, his tips for improving my presentation skills and technical writing style, to mention a few. Sriram has always been a great source of motivation. I cannot forget the hours we spent discussing research ideas. I have also greatly benefited from the weekly group meetings that he has been very particular about. It is the venue where the new graduate students are exposed to what is happening in their field of research, in addition to being a very good platform for losing stage-fright. It has been a great pleasure working with Sanjay and Sriram and I cannot thank them enough for what they have done for me.

Special thanks are due to Prof. Lei Ying from Iowa State University and Prof. R. Srikant from UIUC. I have had a very fruitful collaboration with them during my PhD and I look forward to the same in the future.

I thank Profs. Hasenbein, Srikant, Caramanis and de Veciana for serving on my PhD committee. I am grateful to Profs. de Veciana, Shakkottai, Žitković and Plaxton for being such excellent teachers for hundreds of graduate students like me.

It's been a privilege being part of a premiere research facility - WNCG for the last five years. I have enjoyed working with the top-quality researchers and graduate students. I have had innumerable research discussions with my lab-mates including Aditya Gopalan, Sundar Subramanian, Brian Smith, Siddhartha Banerjee, Zrinka Puljiz, Sriram Sridharan, Aneesh Reddy, Sandeep Bhadra, Amin Jafarian, Jubin Jose, Rajiv Soundararajan, Abhik Das, Shweta Agrawal, Kumar Appaiah, and many others. Without their help this thesis would not have been possible. I am forever indebted to them.

The administrative staff at WNCG over the years - Janet Preuss, Wanda Franklin, Julie Levy, Jocelyn Charvet, Adrian Duran, Paul White deserve special mention and thanks. Their hard work is instrumental in the smooth functioning of the group.

Last but not the least, I express my deepest gratitude to my parents and my sister. Their unconditional support, encouragement and love have been with me throughout this journey.

# High-performance Scheduling Algorithms for Wireless Networks

Publication No. \_\_\_\_\_

Shreeshankar Ravishankar Bodas, Ph.D.  
The University of Texas at Austin, 2010

Supervisors: Sanjay Shakkottai  
Sriram Vishwanath

The problem of designing scheduling algorithm for multi-channel (e.g., OFDM-based) wireless downlink networks is considered, where the system has a large bandwidth and proportionally large number of users to serve. For this system, while the classical MaxWeight algorithm is known to be throughput-optimal, its buffer-overflow performance is very poor (formally, it is shown that it has zero rate function in our setting). To address this, a class of algorithms called iHLQF (iterated Heaviest matching with Longest Queues First) is proposed. The algorithms in this class are shown to be throughput-optimal for a general class of arrival/channel processes, and also rate-function optimal (i.e., exponentially small buffer overflow probability) for certain arrival/channel processes, where the channel-rates are 0 or 1 packets per timeslot. iHLQF however has higher computational complexity than MaxWeight ( $n^4$  vs.  $n^2$  computations per timeslot respectively). To overcome this issue, a

new algorithm called SSG (Server-Side Greedy) is proposed. It is shown that SSG is throughput-optimal, results in a much better per-user buffer overflow performance than the MaxWeight algorithm (positive rate function for certain arrival/channel processes), and has a computational complexity ( $n^2$ ) that is comparable to the MaxWeight algorithm. Thus, it provides a nice trade-off between buffer-overflow performance and computational complexity.

For multi-rate channel processes, where the channels can serve multiple packets per timeslot, new Markov chain-based coupling arguments are used to derive rate-function positivity results for the SSG algorithm. Finally, an algorithm called DMEQ is proposed and shown to be rate-function optimal for certain multi-rate channel scenarios, whose definition characterizes the sufficient conditions for rate-function optimality in this regime. These results are validated by both analysis and simulations.

# Table of Contents

<b>Acknowledgments</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xiv</b>
<b>Chapter 1. Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.1.1 Problem description . . . . .	3
1.2 Main Contributions . . . . .	6
1.3 Related work . . . . .	8
1.4 Organization . . . . .	9
<b>Chapter 2. Iterative Scheduling for Downlink Networks</b>	<b>11</b>
2.1 System Model and Problem Statement . . . . .	11
2.2 Algorithm-Independent Lower Bound on Overflow Probability	15
2.3 Stability and Perfect Matchings . . . . .	16
2.4 Characteristics of Optimal Service Rules . . . . .	19
2.5 A Specific Algorithm: iLQF with PullUp . . . . .	24
2.5.1 Computational Complexity . . . . .	30
2.5.2 Rate-function Optimality . . . . .	30
2.6 Generalizing the System Model . . . . .	33
2.6.1 The Asymmetric Arrivals Case . . . . .	33
2.6.2 Symmetric, Bursty, ON-OFF Arrivals . . . . .	35
2.6.3 Symmetric Arrivals with Bounded Support . . . . .	37
2.6.4 Asymmetric Arrivals with Bounded Support . . . . .	38

<b>Chapter 3. Low-complexity Scheduling Algorithms</b>	<b>40</b>
3.1 iHLQF Analysis . . . . .	45
3.2 The MaxWeight Rule . . . . .	51
3.3 The SSG Scheduling Rule . . . . .	54
3.4 Simulation Results . . . . .	62
<b>Chapter 4. Multi-rate Channel Analysis</b>	<b>68</b>
4.1 Preliminary Analysis . . . . .	71
4.2 Stochastic Dominance of Markov Chains . . . . .	74
4.3 Analysis of the SSG Scheduling Rule . . . . .	77
4.4 The DMEQ Scheduling Rule . . . . .	81
4.4.1 Throughput-optimality of DMEQ . . . . .	86
4.4.2 DMEQ Rate-function, Assumption 4.1 . . . . .	87
4.4.3 DMEQ Rate-function, Assumption 4.1, $M \leq K$ . . . . .	93
4.4.4 DMEQ Rate-function, Assumption 4.2 . . . . .	95
4.5 Unequal Number of Queues and Servers . . . . .	96
<b>Chapter 5. Conclusions</b>	<b>98</b>
<b>Appendices</b>	<b>100</b>
<b>Appendix A. Proofs for Chapter 2</b>	<b>101</b>
A.1 Proof of Theorem 2.2 . . . . .	101
A.2 Proof of Lemma 2.1 . . . . .	102
A.3 Proof of Lemma 2.2 . . . . .	105
A.4 Proof of Lemma 2.3 . . . . .	106
A.5 Proof of Theorem 2.3 . . . . .	107
A.6 Proof of Lemma 2.5 . . . . .	115
A.7 Proof of Lemma 2.6 . . . . .	115
A.8 Proof of Lemma 2.7 . . . . .	117
A.9 Proof of Lemma 2.8 . . . . .	120
A.10 Proof of Lemma 2.9 . . . . .	125
A.11 Proof of Theorem 2.5 . . . . .	126

A.12	Proof of Theorem 2.6 . . . . .	127
A.13	Proof of Theorem 2.7 . . . . .	128
A.14	Proof of Lemma 2.11 . . . . .	128
A.15	Proof of Lemma 2.12 . . . . .	131
A.16	Proof of Theorem 2.8 . . . . .	135
A.17	Proof of Theorem 2.9 . . . . .	137
<b>Appendix B. Proofs for Chapter 3</b>		<b>138</b>
B.1	Proof of Lemma 3.1 . . . . .	138
B.2	Proof of Lemma 3.2 . . . . .	138
B.3	Proof of Lemma 3.3 . . . . .	141
B.4	Proof of Theorem 3.3 . . . . .	142
B.5	Proof of Lemma 3.4 . . . . .	144
B.6	Proof of Lemma 3.5 . . . . .	145
B.7	Proof of Theorem 3.5 . . . . .	146
B.8	Proof of Lemma 3.6 . . . . .	147
B.9	Proof of Lemma 3.8 . . . . .	148
B.10	Proof of Lemma 3.9 . . . . .	150
B.11	Proof of Theorem 3.8 . . . . .	153
<b>Appendix C. Proofs for Chapter 4</b>		<b>155</b>
C.1	Proof of Lemma 4.1 . . . . .	155
C.2	Proof of Theorem 4.1 . . . . .	155
C.3	Proof of Lemma 4.2 . . . . .	156
C.4	Proof of Theorem 4.2 . . . . .	157
C.5	Proof of Lemma 4.3 . . . . .	157
C.6	Proof of Theorem 4.3 . . . . .	159
C.7	Proof of Theorem 4.4 . . . . .	161
C.8	Proof of Theorem 4.5 . . . . .	162
C.9	Proof of Lemma 4.4 . . . . .	164
C.10	Proof of Lemma 4.5 . . . . .	165
C.11	Proof of Theorem 4.6 . . . . .	169
C.12	Proof of Lemma 4.7 . . . . .	172

C.13 Proof of Lemma 4.8 . . . . .	174
C.14 Proof of Lemma 4.9 . . . . .	174
C.15 Proof of Lemma 4.10 . . . . .	175
C.16 Proof of Lemma 4.11 . . . . .	176
C.17 Proof of Lemma 4.12 . . . . .	179
C.18 Proof of Theorem 4.12 . . . . .	179
<b>Bibliography</b>	<b>181</b>
<b>Index</b>	<b>186</b>
<b>Vita</b>	<b>187</b>

# List of Tables

2.1	Notation . . . . .	12
-----	--------------------	----

## List of Figures

1.1	System Model - First Glance . . . . .	3
2.1	System Model . . . . .	12
2.2	Service Model . . . . .	14
2.3	An example of the iLQF algorithm . . . . .	21
2.4	Service model for the queuing system $\mathcal{R}$ . . . . .	23
2.5	An example of the PullUp operation . . . . .	27
3.1	An example of the SSG rule . . . . .	56
3.2	SSG v/s MaxWeight: Bursty, buffer . . . . .	63
3.3	SSG v/s MaxWeight: Bursty, delay . . . . .	63
3.4	SSG v/s Modified iLQF with PullUp: Asymmetric, buffer . . .	64
3.5	SSG v/s Modified iLQF with PullUp: Asymmetric, delay . . .	65
3.6	SSG v/s Modified MaxWeight: Bursty, buffer . . . . .	66
3.7	SSG v/s Modified MaxWeight: Bursty, delay . . . . .	66
3.8	SSG v/s Modified iLQF wuth PullUp: Bursty, buffer . . . . .	67
4.1	Markov Chain for Correlated Channels . . . . .	71
4.2	Candidate Markov chain, Theorem 4.5 . . . . .	77
4.3	Example of a 2-fold perfect matching w.r.t. $\mathcal{U}$ . . . . .	88
A.1	Markov chain for the evolution of the first queue . . . . .	105

# Chapter 1

## Introduction

The last decade has seen an explosion in the types and capabilities of wireless devices deployed all over the world. The demand for the quantity and quality of data transfer via these devices is ever-growing. It is therefore important to understand the fundamental limits on the performance of such wireless systems, and design efficient algorithms and techniques to deliver performance close to the limits.

The traditional approach to the problem of communication over a multi-node network focuses on throughput optimality. Starting with the seminal paper by Tassiulas and Ephremides [30], a number of researchers have contributed to the development of throughput-optimal algorithms in a variety of network models (summarized in Section 1.3). While this work is very important, it largely ignores one important performance metric: *delay*. Especially for voice traffic or video traffic or online gaming, per-user delay is the determining factor for the quality of service. The reason behind this is that while an occasional packet-error or packet-drop is acceptable for real-time traffic, delayed packet delivery is not. It is a goal of this dissertation to design and analyze delay-optimal scheduling algorithms for wireless downlink networks

(see Section 2.1 for more details on the system model).

A significant majority of the present body of work in delay-optimal scheduling (see Section 1.3) falls into two major categories: analysis of average delay or analysis of the tail probabilities of delay, when the queues are long. Some of the techniques used for analyzing these regimes include sample-path large deviations, variational methods, and the bounds on average delay using Lyapunov functions. These results and techniques are instructive for building intuition to design algorithms with good delay performance. Nevertheless, the results are valid in the regime where the queues are large. Therefore, these tools are not directly useful for designing scheduling algorithms for the small-delay regime.

We propose a new framework for analyzing the small delay regime in an appropriate large deviations setting. One of the main differences between this and the previous approaches is that we do not scale time or buffer-size for obtaining delay bounds, but consider the case when the number of users and the available bandwidth is large. Such a setting is expected to be typical in emerging 4G networks. In this setting, we propose to develop scheduling algorithms and develop novel delay analysis techniques based on combinatorial arguments along with large deviations theory.

## 1.1 Motivation

We consider a single-hop wireless downlink network where the base-station has data to transmit to several mobile users. We are motivated by the

anticipated deployment of 4G systems such as WiMax [11] and LTE [1]. These future systems supporting several tens of users at each base-station employ an OFDM<sup>1</sup> based slotted-time air-interface at the base-station. The OFDM air-interface partitions the wireless bandwidth available at the base-station into several hundreds of parallel channels, each of which can be allocated to a (possibly different) user in each time-slot (typically, of the order a few milliseconds). A given band may be allocated to only one user, but a user may be allocated multiple bands. The system model is described in detail in Section 2.1.

### 1.1.1 Problem description

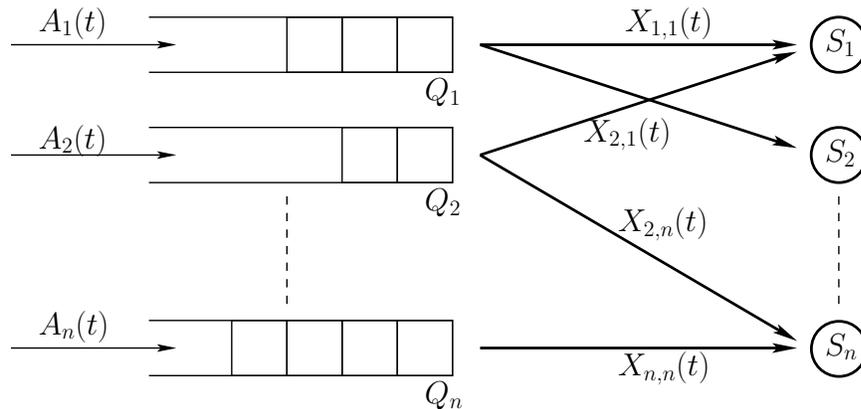


Figure 1.1: System Model - First Glance

To put this section in context, we briefly describe the system model. For more details, please see Section 2.1. Consider a discrete time queuing system with  $n$  queues and  $n$  servers as shown in Figure 2.1. This system

<sup>1</sup>Orthogonal Frequency Division Multiplexing

model can be used to study an OFDM downlink (such as WiMax) where each channel (sub-band), consisting of a fixed number of sub-carriers, is a server in Figure 1.1. There are a fixed number of mobile users, each represented by a queue that corresponds to the backlogged data at the base-station that is destined to the corresponding mobile user. The scheduler operates once every timeslot. During each timeslot, a channel can be assigned to one and at most one user (queue). The state of the channel ( $X_{ij}(t)$ ) to a specific user depends on the location of the user.

Some typical rates (for a 20 MHz WiMax-like system) are as follows: the air-interface is based on OFDM with 50 channels (sub-bands), each of which consists of 25 sub-carriers. Each channel can support 400 kbps and the scheduler operates once every 5 milliseconds. Thus, each good (ON) channel offers 2 kb per timeslot.

We are interested in minimizing the probability that the largest of the queues exceeds a finite number  $b$ , the buffer size. Since the queue-length is directly related to the delay experienced by the user, minimizing the small-buffer overflow probability guarantees high quality of service (QoS) to the users. Now the challenge is to develop a high-performance (low delay) scheduling algorithm for this system. At first glance, by treating each server as a separate downlink server, the problem is not very different from the scheduling for a traditional downlink network. We can then use the following MaxWeight scheduling algorithm, which is throughput-optimal [30]:

**MaxWeight Scheduling:** In timeslot  $t$ , for  $1 \leq j \leq n$ , allocate server  $S_j$  to

serve queue  $Q_i^*$  such that

$$Q_i^* \in \arg \max_{Q_i} X_{ij}(t)Q_i(t),$$

breaking ties arbitrarily. ◇

While the MaxWeight scheduling is throughput-optimal, it causes large delays (due to large queues at the base-station). As an example, assume  $Q_1(t) = 100, Q_2(t) = Q_3(t) = 95, Q_4(t) = Q_5(t) \dots = Q_{100}(t) = 10$ . Then, all servers  $S_j$  such that  $X_{1j}(t) = 1$  will serve  $Q_1$ . Assume that  $X_{ij}(t) = 1$  with probability 0.9, and  $X_{ij}(t)$  are mutually independent. Then, roughly 90% of the servers (channels) will be allocated to user ‘1’, and the remaining 10% to users 2 and 3, which will result in large buffers at users 2 and 3 at the end of the time-slot.

In fact, it can be argued that the MaxWeight algorithm “drives up” all the queue lengths to large enough values to ensure the maximum scheduling flexibility. The reason the MaxWeight algorithm is not the right choice for a scheduling algorithm for this system is that it potentially allocates all the available servers to serve *the* longest queue, essentially treating a slightly smaller queue as if it were empty. For a system with a large number of servers, this allocation policy leads to draining the longest queue(s) much more than is warranted by good load-balancing. It also leads the system getting “trapped” in a state where a significant fraction of queues is long, i.e. once such a state is reached (which happens infinitely often, almost surely, since the system is positive recurrent under the MaxWeight rule), then it is difficult to leave this

state “quickly.” We formalize this observation in Theorem 3.3 to establish that the MaxWeight algorithm results in a very poor delay performance in the multi-channel downlink setting.

A key observation is that for small-buffer multi-channel systems, scheduling needs to be iterative in each time-slot – as resources (channels) get allocated to users, the effect of this allocation needs to be factored in when making allocation decisions for the remaining channels. This kind of an iterative allocation of resources leads to a very good (low) per-user delay performance in addition to network stability.

## 1.2 Main Contributions

1. We present a class of iterative scheduling algorithms called iLQF (iterated Longest Queues First) that is very different from the classic MaxWeight-type algorithms. We show that the algorithms in this class are throughput-optimal for the system (Theorem 3.1), delay-optimal in an appropriate large deviations sense for certain networks (Theorems 3.2, 2.6), and consistently provide a good delay performance under a variety of network conditions (Theorems 2.8, 2.9, 2.11).
2. We show that the classic MaxWeight algorithm results in a very poor delay performance for the wireless downlink system under consideration (Theorems 3.3 and 3.4).
3. We present an algorithm called SSG (Server-Side Greedy) that has the

following properties:

- (a) It is throughput-optimal (Theorem 3.5).
- (b) Its computational complexity per timeslot is comparable to that of the MaxWeight algorithm (Theorems 3.4 and 3.8).
- (c) It provides a very good (but possibly sub-optimal) delay performance in a large deviations sense, under a variety of network settings (Theorems 3.6, 3.7, 4.6, and 4.7).

4. The proofs of the good delay performance of the iLQF and the SSG algorithms use a certain sample-path dominance property that holds only for systems with channel-rates of 0 or 1 packets per timeslot, and fails to hold for even very simple systems with channel-rates of 0 or 2 packets per timeslot. For analyzing systems with multi-rate channels, we present certain properties of Markov chains (Theorems 4.3, 4.4 and 4.5) that are interesting in their own right. We use these properties to prove that the SSG-like iterative algorithms provide a very good delay performance for the systems with multi-rate channels.

5. We ignore the computation complexity considerations and analyze the delay performance of the system with multi-rate channels. We present sufficient conditions for an algorithm to provide the optimal delay performance in certain network settings, and consistently good for various network settings (Section 4.4).

### 1.3 Related work

In their classic paper [30], Tassiulas and Ephremides present a back-pressure scheduling algorithm that is throughput-optimal for a network with arbitrary topology, channel states and interference constraints, which reduces to the MaxWeight algorithm for single-hop flows as considered here. However, the delay performance of this algorithm is well known to be poor, both from an average and a worst-case delay sense (for instance, see [15], [6], [34]). The intuition behind the poor delay performance of the back-pressure algorithm is as follows: the back-pressure algorithm allows the queues to build up to sizable lengths before they are selected for service. This is done in order to ensure maximum scheduling flexibility (in order to ensure guarantee throughput-optimality).

More recently, multiple extensions and alternatives to back-pressure-type algorithms have been developed [31], [2], [27], [24], [10], [14]. Recent progress in studying the performance of scheduling algorithms includes the characterizations of the behavior of queues in heavy-traffic limits [28], [26], [12], [21], [13], computations of the tail probability of queue-lengths using large-deviations analysis [25], [33], [29], [32], and energy-delay tradeoffs for wireless downlink [22]. Order-optimality in number of flows under the MaxWeight algorithm has been explored in [23]. While these results provide very useful insights into the QoS of scheduling algorithms, theoretically, they are valid only when the queue-lengths increase to infinity, i.e., in a large-queue regime.

In a recent work [17], the authors consider a network with primary in-

interference constraints and design a scheduling algorithm that is guaranteed to provide both throughput and delay within a constant factor of the optimal. Their methodology can be extended to more general (secondary/tertiary/...) interference constraints. In [18], the authors consider a model very similar to the one considered in this work, and derive bounds on the expected delays experienced by the users. To the best of our knowledge, the finite buffer analysis presented in this work, for the first time, characterizes the asymptotic buffer overflow performance of OFDM scheduling algorithms in a many-users/servers, small-queue regime.

## 1.4 Organization

In Chapter 2, we define the system model and the problem statement. We present a class of algorithms called iLQF (iterated Longest Queues First) that is optimal for the stated problem and is robust to a variety of changes in the system model.

In Chapter 3, we prove that a natural generalization of the iLQF-class algorithms is throughput-optimal for the system under a very general arrival and channel process. We prove that the MaxWeight algorithm provides a very poor delay performance, and present an iterative algorithm called SSG that provides a trade-off between the delay performance and computational complexity. We show that the SSG algorithm is throughput-optimal. We provide simulation results to compare the different algorithms.

In Chapter 4, we extend the delay analysis to a more general class of

systems with multi-rate channels. We present certain properties of Markov chains that we believe are interesting in their own right and of independent interest. We prove that the SSG-like iterative algorithms provide a good delay performance for these systems. Finally, we characterize sufficient conditions for an algorithm to provide the optimal delay performance for the multi-channel systems under certain network conditions, and conclude in Chapter 5.

## Chapter 2

### Iterative Scheduling for Downlink Networks

In this chapter, we define the system model and the problem statement. We present a class of scheduling algorithms called iLQF (iterated Longest Queues First) as a proposed solution to the problem, show that it is optimal for the problem under consideration, and continues to give consistently good performance under changes to the system model.

#### 2.1 System Model and Problem Statement

We consider a multi-queue, multi-server discrete-time queuing system as shown in Figure 2.1. We first consider a system with equal number of queues and servers,  $n$ .

Table 2.1 summarizes the notation used throughout this dissertation. If no confusion is possible, we denote  $X_{i,j}(t)$  by  $X_{ij}(t)$ . We assume that  $A_i(t), X_{ij}(t), Q_i(t), Q_i^{(k)}(t)$  take values in the set of non-negative integers. The systems are indexed by the number of servers (and queues),  $n$ , are denoted by  $\Upsilon_n$ . (Note that for our proof techniques to work and results to hold, it is not necessary that the number of queues and servers be the same, and a constant factor relationship between the two works just as fine.) For concreteness, we

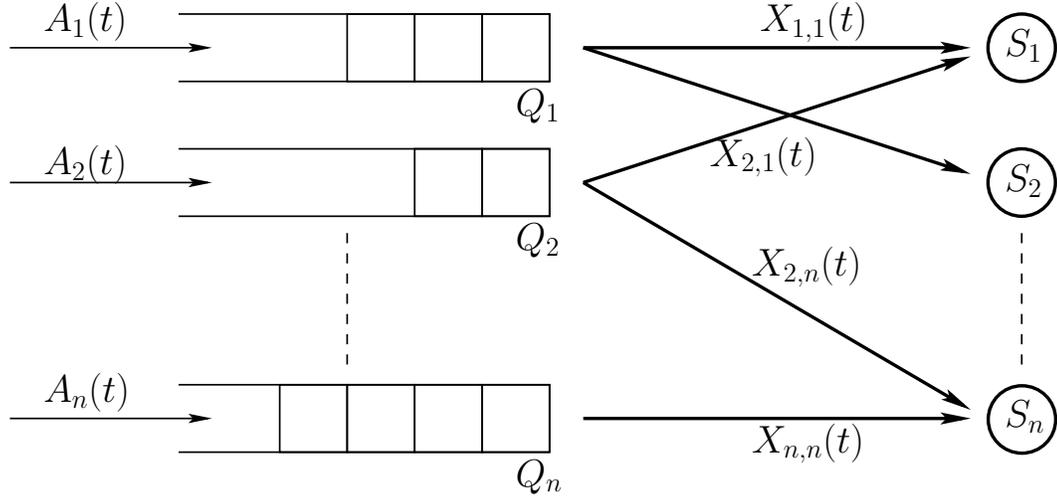


Figure 2.1: System Model

$Q_i$	=	The entity, queue number $i$
$S_i$	=	The entity, server number $i$
$\mathcal{Q}$	=	$\{Q_1, Q_2, \dots, Q_n\}$
$\mathcal{S}$	=	$\{S_1, S_2, \dots, S_n\}$
$A_i(t)$	=	The number of packet arrivals to $Q_i$ at the beginning of timeslot $t$
$X_{i,j}(t)$	=	The number of packets in $Q_i$ that can potentially be served by $S_j$ , in timeslot $t$
$Q_i(t)$	=	The length of $Q_i$ at the end of timeslot $t$
$Q_i^{(k)}(t)$	=	The length of $Q_i$ after $k \geq 1$ rounds of service in timeslot $t$
$Q_i^{(0)}(t)$	=	$Q_i(t-1) + A_i(t)$ , i.e. the length of $Q_i$ after immediately after arrivals, in timeslot $t$
$a^+$	=	$\max(a, 0)$
$\mathfrak{R}_+$	=	The set of nonnegative real numbers
$\mathbb{Z}_+$	=	The set of nonnegative integers
$H(x y)$	=	$x \log \frac{x}{y} + (1-x) \log \frac{1-x}{1-y}$
w.p.	=	with probability
$\mathcal{M}_1(\Sigma)$	=	The probability simplex in $\mathfrak{R}^k$ for appropriate $k$

Table 2.1: Notation

assume that in a given timeslot, there are first arrivals to the queues (if any), then possible service, and the queue-lengths are measured at the end of the timeslot. The arrivals to the queues, and the channels connecting the queues to the servers are i.i.d. Bernoulli, independent across queues and time.<sup>1</sup> In particular,

$$A_i(t) = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1 - p, \end{cases} \quad (2.1.1)$$

and

$$X_{ij}(t) = \begin{cases} 1 & \text{with probability } q, \\ 0 & \text{with probability } 1 - q, \end{cases} \quad (2.1.2)$$

for some  $p, q \in (0, 1)$ . All the random variables  $A_i(t)$  and  $X_{jk}(s)$  are mutually independent for all possible values of the involved parameters. Each queue maintains a buffer of infinite size, so that no packets are ever dropped. If  $X_{ij}(t) = 1$ , then the server  $S_j$  can potentially serve queue  $Q_i$  in timeslot  $t$ , reducing the length of  $Q_i$  by 1 (unless it is empty). Our aim is to define a service rule for allocating the servers to the queues this system, that meets certain performance metrics to be defined. The service rule is allowed to use the entire history of queue-lengths, arrivals, channel realizations, and server allocation decisions, as well as the queue-lengths, channel realizations and arrivals in the current timeslots, and any amount of external randomness (if necessary), and is required to define the following random variables for each

---

<sup>1</sup>We adopt such a system model for ease of exposition. Significant generalization of this model is possible, as noted in Section 2.6.

timeslot  $t$  :

$$Y_{i,j}(t) = \begin{cases} 1 & \text{if } S_j \text{ is allocated to serve } Q_i \text{ in timeslot } t, \\ 0 & \text{otherwise.} \end{cases}$$

If no confusion is possible, we denote  $Y_{i,j}(t)$  by  $Y_{ij}(t)$ . We impose the condition that in a given timeslot, a given server can be allocated to serve at most one queue. This condition translates to the following: for all  $t$  and all  $j \in \{1, 2, \dots, n\}$ , any valid service policy must obey  $\sum_{i=1}^n Y_{ij}(t) \leq 1$ . Thus, the queue-length evolution is given by

$$Q_i(t) = \left( Q_i(t-1) + A_i(t) - \sum_{j=1}^n X_{ij}(t)Y_{ij}(t) \right)^+.$$

The service model is as shown in Figure 2.2.

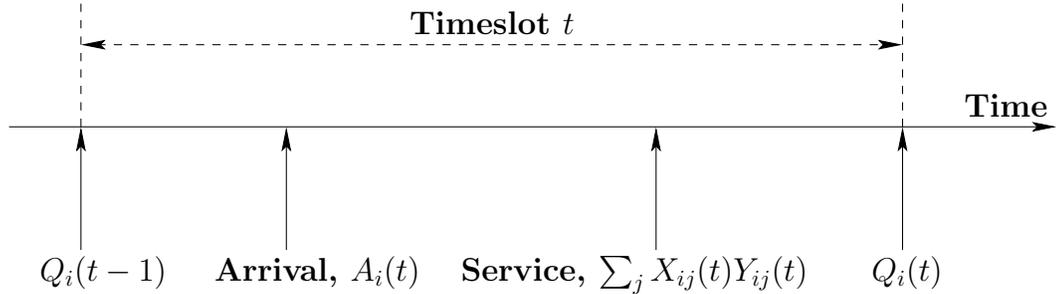


Figure 2.2: Service Model

A finite integer  $b \geq 0$  is fixed. The queuing system is started at time  $-\infty$ . Our objective is to design a service rule that maximizes

$$I(b) := \liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right). \quad (2.1.3)$$

We refer to the event  $\{\max_i Q_i(t) > b\}$  as the *small buffer overflow event* or simply the *overflow event*. The probability term in the above expression can

thus be thought of as the probability of the overflow event under the stationary distribution of the queue-length process (provided one exists). The function  $I(b)$  is called the rate function in the large deviations theory. If a scheduling algorithm results in a positive value of  $I(b)$ , then for large values of  $n$ , we have

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \lesssim e^{-nI(b)},$$

where the inequality holds up to sub-exponential multiplicative factors on the RHS. Thus, the probability of the small buffer overflow event decays to zero very rapidly with  $n$ , and it is desirable to have as large a value of  $I(b)$  as possible.

## 2.2 Algorithm-Independent Lower Bound on Overflow Probability

In this section, we present a lower bound on the overflow probability (Equation (2.1.3)). This is an algorithm-independent lower bound, so it holds for any scheduling algorithm. In Section 2.4, we develop a class of iterative algorithms (iLQF) that achieve this bound.

**Theorem 2.1.** *For the system  $\Upsilon_n$ , under any rule for allocating servers to queues, and for all possible values of the parameters  $n > 0, 0 < p, q < 1, b \geq 0$ ,*

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \geq p^{b+1}(1-q)^{n(b+1)}.$$

*Consequently, for any  $p > 0$ ,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \leq (b+1) \log \frac{1}{1-q}. \quad (2.2.1)$$

*Proof.* Consider the following event which implies  $\{Q_1(0) > b\}$  under any scheduling rule: for  $b + 1$  consecutive timeslots before (and including) timeslot 0, there are arrivals to  $Q_1$ , and all the channels connecting  $Q_1$  to the servers are OFF in each of the  $b + 1$  timeslots. The probability of this event is equal to  $p^{b+1}((1 - q)^n)^{b+1}$ , and the result follows.  $\square$

### 2.3 Stability and Perfect Matchings

In this section, we first show that the system under consideration is stabilizable under some scheduling rule, for  $n$  large.

**Theorem 2.2.** *For given values of  $p, q \in (0, 1)$ , there exists  $n_0 = n_0(p, q)$  such that for all  $n \geq n_0$ , the queuing system  $\Upsilon_n$  can be stabilized by some service rule.*

*Proof.* Please see Appendix A.1.  $\square$

Let  $\lambda_i^{(n)}$  denote the arrival rate (expected number of arrivals) to queue  $Q_i$ , in the system  $\Upsilon_n$ . Assuming that

$$\limsup_{n \rightarrow \infty} \max_{1 \leq i \leq n} \lambda_i^{(n)} \in (0, 1),$$

the above stability result can be generalized to the following cases:

1. Bernoulli arrivals to the queues, with arrival rates to the different queues being different.

2. The number of arrivals to a queue in a given timeslot takes on values in a finite, non-negative integer set.

We analyze the two cases mentioned above in Section 2.6. Next we prove a result regarding perfect matchings in bipartite graphs that is useful for the analysis of the proposed algorithm in Section 2.5.

**Lemma 2.1.** *Consider an undirected bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ , where  $\mathcal{U} \cup \mathcal{V}$  is the set of vertices with  $|\mathcal{U}| = |\mathcal{V}| = n$ , and  $\mathcal{E}$  is the set of edges. Every edge  $e \in \mathcal{E}$  has one of its endpoints in  $\mathcal{U}$  and the other in  $\mathcal{V}$ . For every node  $u \in \mathcal{U}$  and  $v \in \mathcal{V}$ , the edge  $(u, v)$  is present in  $\mathcal{E}$  with probability  $q$ , independently of all other edges. Then, for large  $n$ ,*

$$(1 - q)^n \leq \mathbb{P}(G \text{ has no perfect matching}) \leq 3n(1 - q)^n,$$

where a perfect matching is defined as a matching of cardinality  $n$ .

*Proof.* Please see Appendix A.2. □

Qualitatively, this result shows that the large bipartite graphs as described here have perfect matchings with very high probability.

Next we consider a perfect-matching scheduling. This simple, queue-length-agnostic scheduling rule is rate-function optimal for the problem under consideration, but is sensitive to the nature of the arrival process, as the following exposition makes clear.

**Definition 2.1** (Perfect-matching scheduling). *In a timeslot  $t$ , let  $\mathcal{E} := \{X_{ij}(t) : X_{ij}(t) = 1\}$ . If there exists a perfect matching in the bipartite graph  $G(\mathcal{Q} \cup \mathcal{S}, \mathcal{E})$ , then allocate the servers to serve the respective queues as determined by the perfect matching, else do not allocate any server to the queues.  $\diamond$*

**Lemma 2.2.** *For the system  $\Upsilon_n$ , the perfect-matching scheduling yields*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq (b + 1) \log \frac{1}{1 - q}.$$

*Thus, in conjunction with (2.2.1), the perfect matching scheduling rule maximizes (2.1.3), and is rate-function optimal.*

*Proof.* Please see Appendix A.3.  $\square$

While the perfect-matching scheduling is rate-function optimal for  $A_i(t) \in \{0, 1\}$ , this algorithm is sensitive to the arrival processes.

**Definition 2.2.** *The arrival process to a queuing system is said to be  $L \times$  Bernoulli( $p$ ) if it satisfies*

$$A_i^{(n)}(t) = \begin{cases} L & \text{with probability } p, \\ 0 & \text{with probability } 1 - p, \end{cases}$$

*with  $pL < 1$ . If  $L = 1$ , then the process is said to be Bernoulli( $p$ ).  $\diamond$*

**Lemma 2.3.** *If the arrival process to the system  $\Upsilon_n$  is changed from Bernoulli( $p$ ) to  $2 \times$  Bernoulli( $r$ ) for any  $r \in (0, 0.5)$ , then the perfect matching scheduling rule results in the overflow event having at least a constant probability, implying that the expression (2.1.3) equals 0.*

*Proof.* Please see Appendix A.4. □

This motivates us to study (in the rest of this chapter) a queue-length based scheduling policy which provides rate-function optimality (in Equation (2.1.3)) for the Bernoulli( $p$ ) arrival process, and also achieves a nonzero rate function for more general arrival processes.

## 2.4 Characteristics of Optimal Service Rules

In this section, we consider a special class of service rules - iLQF (iterated Longest Queues First), and present sufficient conditions for an iLQF scheduling policy to be rate-function optimal. In the next section, we present an algorithm in this class that maximizes (2.1.3).

**Definition 2.3** (iterated Longest Queues First (iLQF)). *A service rule is said to belong to the class iLQF if, in every timeslot, it allocates servers to queues in multiple rounds as follows:*

1. *In a given round, the service rule finds a largest cardinality matching in the bipartite graph whose node-sets are the set of longest queues and set of available servers, and the edges are defined by the channel realizations (an edge from  $Q_i$  to  $S_j$  is present if  $X_{ij} = 1$ ), and allocates the servers to the (longest) queues as determined by the matching. If the cardinality of the matching thus found equals the cardinality of the set of the longest queues, then the algorithm is required to serve all the queues. If, in the given round, none of the longest queues are connected to any of the*

servers, then the set of the next longest queues may be considered for server allocation, but it is not required to be considered.

2. The service rule updates the lengths of all the queues (to take into account the service received by a subset of the longest queues in the particular round) and the set of available servers (to take into account the servers allocated to some of the queues) and proceeds to the next round.  $\diamond$

Note that the class iLQF contains more than one scheduling algorithm, since the following parameters are unspecified:

1. The number of rounds to be performed, i.e. the termination condition.
2. The tie-breaking rule if there exist multiple largest cardinality matchings among the longest queues.

Consider an algorithm in the iLQF class that arbitrarily breaks ties in the case of multiple largest matchings, and terminates if none of the longest queues can be served in a given round. A typical execution of this iLQF algorithm is shown in Figure 2.3.

This class of algorithms is interesting because it gives priority to the longer queues, thereby trying to minimize the probability of the overflow event.

**Lemma 2.4.** *For any algorithm in the iLQF class, and for  $n$  large enough,*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(t+1) > \max_{1 \leq i \leq n} Q_i(t) \right) \leq 3n(1-q)^n.$$

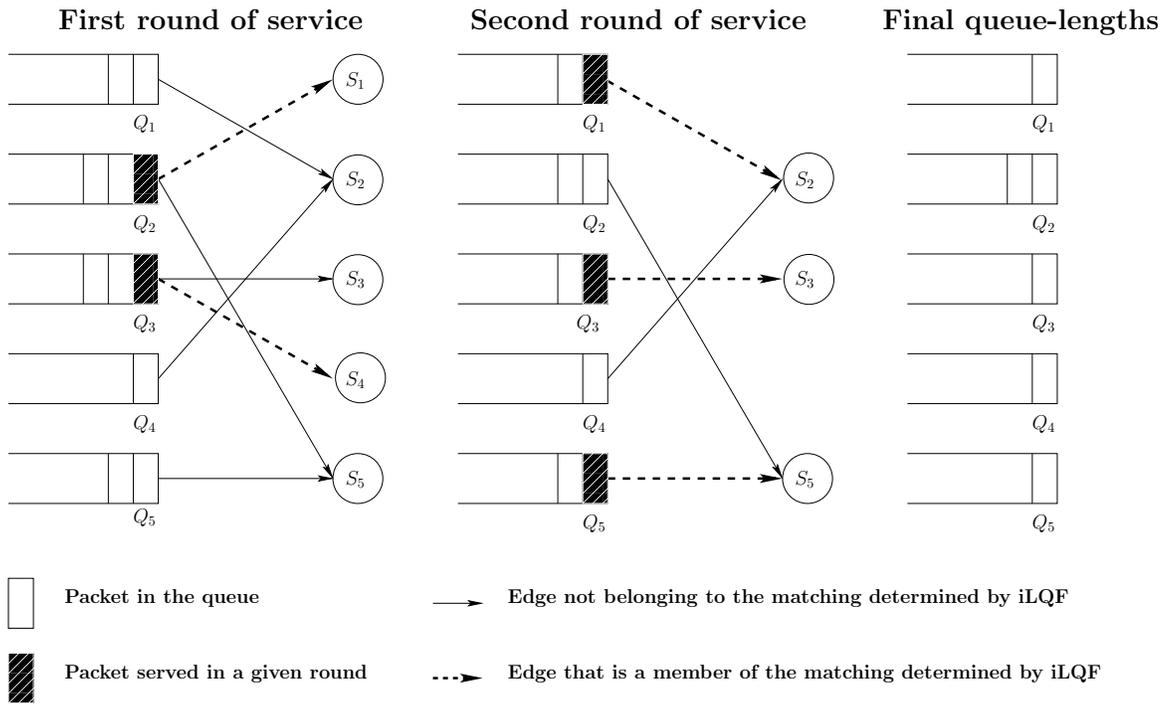


Figure 2.3: An example of the iLQF algorithm

*Proof.* Consider the bipartite graph  $G(\mathcal{Q} \cup \mathcal{S}, \mathcal{E})$ , where  $\mathcal{E} := \{X_{ij}(t) : X_{ij}(t) = 1\}$ . If  $G$  has a perfect matching (i.e., a matching of cardinality  $n$ ), then for an algorithm in the iLQF class,  $\max_{1 \leq i \leq n} Q_i(t+1) \leq \max_{1 \leq i \leq n} Q_i(t)$ . Further, by Theorem 2.1, the graph  $G$  has a perfect matching with probability at least  $1 - 3n(1-q)^n$  for large  $n$ .  $\square$

In order to establish the rate-function optimality of the iLQF algorithms, we restrict our attention to a subset of the iLQF-class, whose elements possess two key properties: the *Drain* property and *Dominance* property. In Section 2.5, we present a particular iLQF-class algorithm which possesses these properties. We first define the Dominance property.

**Definition 2.4** (Dominance property of an iLQF rule  $\Lambda$ ). *Consider the queuing system with  $\mathcal{Q} = \{Q_i\}_{i=1}^n$  as the queues, and  $\mathcal{S} = \{S_i\}_{i=1}^n$  as the servers. Let  $A_i(t)$  and  $X_{ij}(t)$  be the arrival process and channel processes respectively (see Equations (2.1.1), (2.1.2)). Now, a new queuing system with queues  $\mathcal{R} = \{R_i\}_{i=1}^n$  and servers  $\mathcal{S} = \{S_i\}_{i=1}^n$  is obtained as follows: at each time  $t$ , the queues  $R_i(t), i = 1, 2, \dots, n$  see the same arrivals as those incoming to  $Q_i(t), i = 1, 2, \dots, n$  and the channel states of the servers are identical to those of system  $\mathcal{Q}$  (i.e., the arrival processes and channel states in the system  $\mathcal{R}$  are sample-path coupled with the system  $\mathcal{Q}$ ). In addition, there are extra packet arrivals (an arbitrary, finite number) that occur to an arbitrary subset of queues in the system  $\mathcal{R}$  immediately after service, and at arbitrary timeslots  $T_1, T_2, \dots$  (see Figure 2.4). The service policy used in the queuing system  $\mathcal{R}$  is the same*

*iLQF* policy ( $\Lambda$ ) that is used in the system  $\mathcal{Q}$  (also the process  $\mathcal{R}$  is defined over the same probability space as  $\mathcal{Q}$ ).

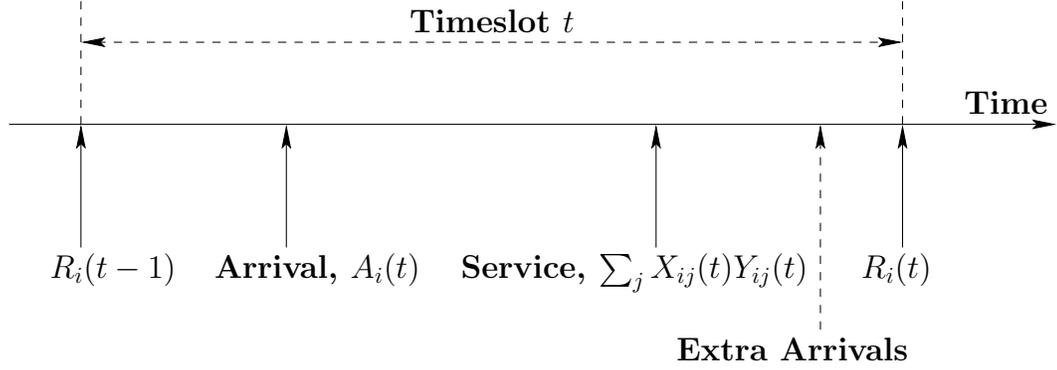


Figure 2.4: Service model for the queuing system  $\mathcal{R}$

A rule  $\Lambda$  in the *iLQF* class is said to have the dominance property if the following holds: for all timeslots  $t$ , all  $b \geq 0$ , and over all possible choices for extra arrivals,

$$\mathbb{P} \left( \max_{1 \leq i \leq n} R_i(t) > b \right) \geq \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(t) > b \right). \quad \diamond$$

Intuitively, the dominance property requires that *adding extra packets* to the queueing system driven by the *iLQF* policy  $\Lambda$  does *not decrease* the maximum queue length. This property is extremely useful, because this property allows us to “carefully” add packets so that the resulting queueing system can be explicitly analyzed and whose rate function can be computed in closed-form. The dominance property ensures that the rate function so obtained provides a lower-bound on the rate-function of the original system.

**Definition 2.5** (Drain property of a scheduling rule  $\Lambda$ ). *A scheduling rule  $\Lambda$  (not necessarily from the iLQF-class) is said to have the drain property if there exists a constant  $k_0$  independent of  $n$  such that*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(t + k_0) < \max_{1 \leq i \leq n} Q_i(t) \mid \max_{1 \leq i \leq n} Q_i(t) > 0 \right) \geq \frac{1}{2},$$

for all  $n$  large enough and all integers (timeslots)  $t$ . ◇

In words,  $k_0$  is an integer such that the maximum queue length decreases in  $k_0$  timeslots with probability greater than or equal to  $1/2$ . We are now ready to state the main theorem of this section.

**Theorem 2.3.** *Suppose a service rule in the iLQF class has the drain and dominance properties. Then, this iLQF service rule results in*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = (b + 1) \log \frac{1}{1 - q}.$$

Further, by Theorem 2.1, no other service rule can give a larger value for the left hand side of the above expression.

*Proof.* Please see Appendix A.5. □

## 2.5 A Specific Algorithm: iLQF with PullUp

We now focus our attention on constructing an algorithm in the iLQF class that satisfies the requirements in the statement of Theorem 2.3. The algorithm employs a particular tie-breaking rule (PullUp) when there exist multiple largest-cardinality matchings in the bipartite graph between the set of queues and servers, where the edges are defined by the ON links.

Before we explain the intuition behind this tie-breaking rule, consider two queuing systems denoted by  $\mathcal{Q}$  and  $\mathcal{R}$  with both these systems operating with the same iLQF rule. Further, suppose that at some timeslot along a fixed sample-path, the set of longest-queues under  $\mathcal{Q}$  is a subset of the set of longest queues under  $\mathcal{R}$  and the set of available channels in system  $\mathcal{Q}$  are “more” than that in system  $\mathcal{R}$  (more precisely, the bipartite graph connecting the queues to the servers in system  $\mathcal{Q}$  has more servers and edges than in the system  $\mathcal{R}$ , where ordering is defined by set inclusion). This is a scenario where system  $\mathcal{R}$  is less “flexible” than system  $\mathcal{Q}$  (in terms of scheduling flexibility) in the sense that any allocation of servers in the system  $\mathcal{R}$  can be mimicked by system  $\mathcal{Q}$ .

Now, the intuition behind PullUp can be explained as follows: consider two multichannel queuing systems with identical initial conditions and suppose we add packets at arbitrary times to one of them (say, the second system). Then, we would like the first system to have more “flexibility” at each time slot under iLQF in the sense of the previous paragraph. (We will see that such a property is key to showing the stochastic dominance in Theorem 2.3.) The PullUp-based iLQF algorithm described below ensures that such a property holds.

**Definition 2.6** (PullUp). *Consider a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ , where the sets of nodes, not necessarily of the same cardinality, are  $\mathcal{U} = \{u_1, u_2, \dots, u_m\}$  and  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ . Given a matching  $\mathcal{M}$  in  $G$ ,  $\mathcal{M}' := \text{PullUp}(G, \mathcal{M}, \mathcal{V})$  is a new matching obtained by the following steps, which we call PullUp:*

1. Mark all the edges in  $\mathcal{M}$  as forward edges (i.e. from  $\mathcal{U}$  to  $\mathcal{V}$ ), and all the other edges in  $\mathcal{E}$  as backward edges, to get a directed graph  $G_1$ . Define  $\mathcal{M}_1 := \mathcal{M}$ . Initialize  $k = 1$ .
2. Obtain  $\mathcal{M}_{k+1}$  from  $\mathcal{M}_k$  as follows: If the node  $v_k$  has an incoming edge, then define  $\mathcal{M}_{k+1} := \mathcal{M}_k$ ,  $G_{k+1} := G_k$ . Otherwise, in the directed graph  $G_k$ , find the set  $\mathcal{N}_k$  of all nodes reachable from  $v_k$ . Let  $\Gamma(G_k, v_k) := \mathcal{N}_k \cap \mathcal{U}$  and  $\Delta(G_k, v_k) := \mathcal{N}_k \cap \mathcal{V}$ . Find the smallest index  $l > k$  such that  $v_l \in \Delta(G_k, v_k)$ . If no such  $l$  exists, then define  $\mathcal{M}_{k+1} := \mathcal{M}_k$  and  $G_{k+1} := G_k$ . If such an  $l$  exists, then reverse the directions of all the edges on a path from  $v_k$  to  $v_l$ , to obtain a graph  $G_{k+1}$ . Define  $\mathcal{M}_{k+1}$  to be the set of all forward edges in  $G_{k+1}$ .
3. Increment  $k$  by 1. If  $k = n + 1$ , then return the matching  $\mathcal{M}' := \mathcal{M}_{n+1}$ , else go to step 2.  $\diamond$

An example of the PullUp operation is shown in Figure 2.5. In the next lemma, we prove that the output of the PullUp is also a matching.

**Lemma 2.5.** *The output  $\mathcal{M}'$  of  $\text{PullUp}(G, \mathcal{M}, \mathcal{V})$  is a matching, and  $|\mathcal{M}| = |\mathcal{M}'|$ .*

*Proof.* Please see Appendix A.6.  $\square$

The objective of the PullUp operation is to efficiently find a “good” matching. Based on the PullUp technique, we construct an iLQF-class algo-

rithm that is rate-function optimal for the small buffer overflow event under consideration.

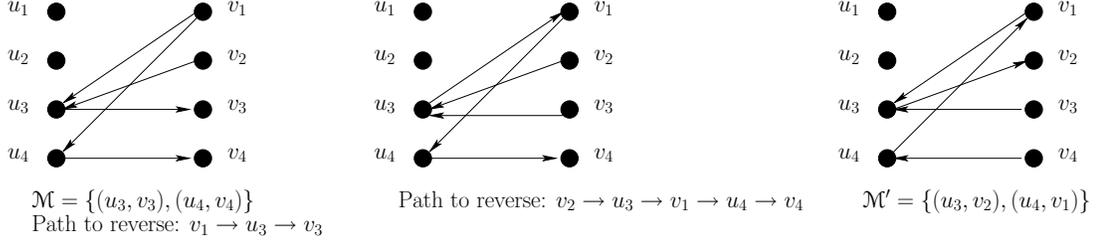


Figure 2.5: An example of the PullUp operation

**Definition 2.7** (iLQF with PullUp).

**Input:**

1. The queue lengths,  $Q_1(t-1), Q_2(t-1), \dots, Q_n(t-1)$ .
2. The channel realizations,  $X_{ij}(t)$  for  $1 \leq i, j \leq n$ .
3. The arrivals to the queues,  $A_i(t)$  for  $1 \leq i \leq n$ .

**Steps:**

1. Update the queue-lengths to account for arrivals, that is, compute the new length of queue  $Q_i$  after arrivals,  $Q_i^{(0)}(t) := Q_i(t-1) + A_i(t)$ . Hereafter, the length of a queue always refers to its most current updated length, accounting for arrivals and service. Find the length of the longest queue,  $\hat{Q}$ . Define  $L = \hat{Q}$ . Initialize  $r = 0$ . Let  $\mathcal{S}^*$  denote the set of unallocated servers. To begin with, we have  $\mathcal{S}^* = \mathcal{S}$ .

2. Let  $\mathcal{Q}_L$  denote the set of queues whose length (i.e.,  $Q_i^{(r)}(t)$ ) is exactly  $L$ . Let  $G_L$  denote the (undirected) bipartite graph with nodes  $\mathcal{Q}_L \cup \mathcal{S}^*$ , and the edges as defined by the channel realizations. More specifically, an edge  $(Q_i, S_j)$  is present in  $G_L$  if  $Q_i \in \mathcal{Q}_L, S_j \in \mathcal{S}^*$  and  $X_{ij}(t) = 1$ . Find a largest cardinality matching  $\mathcal{M}_L$  in the graph  $G_L$ .

(a) If  $|\mathcal{M}_L| = |\mathcal{Q}_L|$ , then define  $\mathcal{M}' := \text{PullUp}(G_L, \mathcal{M}_L, \mathcal{S}^*)$ .

(b) If  $|\mathcal{M}_L| < |\mathcal{Q}_L|$ , then define  $\mathcal{M}_1 := \text{PullUp}(G_L, \mathcal{M}_L, \mathcal{S}^*)$ . Obtain  $\mathcal{M}_{k+1}$  from  $\mathcal{M}_k$  as follows: if  $k$  is odd, then define  $\mathcal{T} := \mathcal{Q}_L$ ; otherwise  $\mathcal{T} := \mathcal{S}^*$ , and  $\mathcal{M}_{k+1} := \text{PullUp}(G_L, \mathcal{M}_k, \mathcal{T})$ . Continue obtaining  $\mathcal{M}_{k+1}$  from  $\mathcal{M}_k$  until  $\mathcal{M}_{i+1} = \mathcal{M}_i$  for some  $i$ . Define  $\mathcal{M}' := \mathcal{M}_i$ .

Finally, as defined by the matching  $\mathcal{M}'$ , allocate the servers to queues. For example, if  $(Q_x, S_y) \in \mathcal{M}'$ , then define  $Y_{xy}(t) = 1$ , allocate  $S_y$  to serve  $Q_x$ , remove  $S_y$  from  $\mathcal{S}^*$ , decrease the length  $Q_x$  by 1, i.e., define  $Q_x^{(r+1)}(t) := Q_x^{(r)}(t) - 1$ . For a node (queue)  $Q_z$  that is not an endpoint of any edge in  $\mathcal{M}'$ , define  $Q_z^{(r+1)}(t) := Q_z^{(r)}(t)$ .

3. If at the end of step 2, we have  $|\mathcal{M}_L| < |\mathcal{Q}_L|$ , then stop. If  $|\mathcal{S}^*| = 0$  or  $L = 1$ , then stop. Else, decrease the value of  $L$  by 1, increment  $r$  by 1, go to step 2.

**Output:**

1. The allocation decisions,  $Y_{ij}(t)$  for  $1 \leq i, j \leq n$ .

2. The final queue-lengths,  $Q_i(t) := Q_i^{(r+1)}(t)$ . (Here, the value of  $r$  refers to its value at the end of step 3.)  $\diamond$

Here is a description of the algorithm in words: in every timeslot, the algorithm proceeds in multiple rounds of service. In every round, the algorithm finds a largest-cardinality matching  $\mathcal{M}$  in the (bipartite) graph defined by the longest queues and the unallocated servers, where the edges in the graph are defined by the channel realizations. The algorithm then applies the PullUp operation (once or multiple times) to the matching  $\mathcal{M}$  and obtains a matching  $\mathcal{M}'$ . It allocates servers to the queues as defined by the edges in the matching  $\mathcal{M}'$ , updates (decreases by one) the lengths of the served queues, removes the allocated servers from the set of available servers, and proceeds to the next round.

Let every execution of step 2 be called a round. If in the step 2 we have  $|\mathcal{M}_L| = |\mathcal{Q}_L|$ , then that round is called a perfect matching round, else a maximal matching round.

**Theorem 2.4.** *The iLQF with PullUp is rate-function optimal, i.e., it results in*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = (b+1) \log \frac{1}{1-q}.$$

*Further, the algorithm can be implemented in  $O(n^4)$  computations per timeslot.*

*Proof.* We prove that the iLQF with PullUp satisfies the drain property (Lemma 2.10) and the dominance property (Lemma 2.8). Thus, the first part of the claim

holds according to Theorem 2.3. The second part of the claim (the computational complexity result) follows from Lemma 2.6.  $\square$

### 2.5.1 Computational Complexity

We first analyze the computational complexity of the iLQF with PullUp.

**Lemma 2.6.** *The proposed algorithm (iLQF with PullUp) can be implemented in  $O(n^4)$  computations per timeslot.*

*Proof.* Please see Appendix A.7.  $\square$

### 2.5.2 Rate-function Optimality

We establish the rate-function optimality of the iLQF with PullUp by proving that the algorithm has the drain property and the dominance property as required by Theorem 2.3. The following is a technical lemma that is useful in the proof of Lemma 2.8.

**Lemma 2.7.** *In the graph  $G_{n+1}$ , if a node  $v_a$  has no incoming edge, then there does not exist a (directed) path from  $v_a$  to any node  $v_b$  with  $b > a$ . Consequently, if  $\text{PullUp}(G, \mathcal{M}, \mathcal{V}) = \mathcal{M}'$ , then  $\text{PullUp}(G, \mathcal{M}', \mathcal{V}) = \mathcal{M}'$ .*

*Proof.* Please refer to Appendix A.8.  $\square$

**Lemma 2.8** (Sample-path-wise Dominance). *Consider two queuing systems  $\underline{Q}$  and  $\underline{R}$  with queues  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$  and  $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$  respectively, with the property that  $Q_i(t-1) \leq R_i(t-1)$  for all  $i$ . Let the two*

systems have identical channel realizations,  $X_{ij}(t)$  and identical arrivals,  $A_i(t)$  for  $1 \leq i, j \leq n$ . Both the queuing systems implement the algorithm described in Section 2.5, i.e. *iLQF with PullUp*. Then,  $Q_i(t) \leq R_i(t)$  for all  $i$ .

*Proof.* Please see Appendix A.9. □

Note that this theorem immediately implies that the *iLQF with PullUp* algorithm has the dominance property as required by Theorem 2.3.

**Corollary 2.1.** *The iLQF with PullUp algorithm has the dominance property defined in Section 2.4.*

The corollary follows by repeated applications of Theorem 2.8. The queuing system is started at time  $-\infty$ , and we are interested in the probability that the length of the longest queue exceeds a constant  $b$  at a finite time  $t$ . By applying the result of Theorem 2.8 to timeslots  $T_1, T_2, \dots$  (in the definition of the Dominance property), it follows that the packet-added system has sample-path wise longer queues than the original system. The probabilistic dominance is an immediate consequence of this sample-path dominance.

We now demonstrate a property of the PullUp operation which is useful in proving that the proposed algorithm has the Drain property as required by Theorem 2.3.

**Lemma 2.9.** *Let a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$  and a matching  $\mathcal{M}$  be given, with*

$$\mathcal{U} = \{u_1, u_2, \dots, u_n\}, \mathcal{V} = \{v_1, v_2, \dots, v_n\}.$$

Suppose there exists a matching  $\mathcal{M}_\star$  in  $G$  with the following properties:

1.  $|\mathcal{M}| = |\mathcal{M}_\star|$ .
2. If  $u \in \mathcal{U}$  is an endpoint of some edge  $e \in \mathcal{M}$ , then  $u$  is an endpoint of some edge  $e' \in \mathcal{M}_\star$ .
3. Mark all the edges in  $\mathcal{M}_\star$  as forward edges (i.e., from  $\mathcal{U}$  to  $\mathcal{V}$ ), and all the edges in  $\mathcal{E} \setminus \mathcal{M}_\star$  as backward edges, to get a directed graph  $G^\ddagger(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ . Then, in the graph  $G^\ddagger$ , if a node  $v_i \in \mathcal{V}$  has no incoming edge, then there does not exist a directed path from  $v_i$  to any  $v_j$ ,  $j > i$ .
4. For some  $a \leq n$  and every  $b > a$ , no node  $v_b \in \mathcal{V}$  is an endpoint of any edge in  $\mathcal{M}_\star$ .

Let  $\mathcal{M}' = \text{PullUp}(G, \mathcal{M}, \mathcal{V})$ . Then, any edge in  $\mathcal{M}'$  does not have, as an endpoint, any node in  $\mathcal{V}$  with index larger than  $a$ .

*Proof.* Please refer to Appendix A.10. □

Let  $\ell_T$  denote the length of the longest queue at the end of the timeslot  $T$ . We next prove that the iLQF with PullUp satisfies the drain property.

**Lemma 2.10.** (*The Drain property*) For the proposed algorithm, there exists a constant  $k = k(p) = \left\lceil \frac{3}{1-p} \right\rceil$  such that, for all  $n$  large enough, all  $m > 0$  and all  $T$ ,

$$\mathbb{P}(\ell_{T+k} < m | \ell_T = m) \geq \frac{1}{2}.$$

*Proof.* Please refer to Lemma 2.12, which proves a more general claim. Substituting  $L = 1$  in that lemma gives the desired result.  $\square$

This completes our analysis of the iLQF with PullUp algorithm for the basic system model,  $\Upsilon_n$ . In conclusion, we have shown that the proposed algorithm - iLQF with PullUp is rate-function optimal for the small buffer overflow event under consideration.

## 2.6 Generalizing the System Model

In this section, we consider a number of natural extensions of the system model  $\Upsilon_n$  defined in Section 2.1, and analyze the performance of the proposed iLQF with PullUp algorithm for them.

### 2.6.1 The Asymmetric Arrivals Case

Consider a queuing system  $\Upsilon'_n$  that is a modification of the system  $\Upsilon_n$ . Let the queues, servers, arrivals and channels for the system  $\Upsilon'_n$  be indexed by  $n$  and denoted by  $Q_i^{[n]}, S_j^{[n]}, A_i^{[n]}(t)$  and  $X_{ij}^{[n]}(t)$  respectively. Let

$$A_i^{[n]}(t) = \begin{cases} 1 & \text{with probability } p_i^{(n)}, \\ 0 & \text{with probability } 1 - p_i^{(n)}, \end{cases} \quad (2.6.1)$$

and

$$X_{ij}^{[n]}(t) = \begin{cases} 1 & \text{with probability } q, \\ 0 & \text{with probability } 1 - q. \end{cases}$$

In particular, the number of packets arriving to the  $i^{th}$  queue in timeslot  $t$  in the system  $\Upsilon'_n$  is a Bernoulli random variable whose parameter is arbitrarily

fixed in  $(0, 1)$ . For stability, we impose the condition

$$\limsup_{n \rightarrow \infty} \max_{1 \leq i \leq n} p_i^{(n)} = \alpha \in (0, 1). \quad (2.6.2)$$

Under this condition, following an argument similar to that in the proof of Theorem 2.2, the system  $\Upsilon'_n$  is stable for all  $n$  large enough. We refer to this system as a system with asymmetric arrivals.

**Theorem 2.5.** *For any given  $\epsilon \in (0, \alpha)$ , there exists a constant  $n_0 = n_0(\epsilon)$  such that under any rule for allocating servers to queues, and for all possible values of the parameters  $0 < \alpha, q < 1, b \geq 0$ , and for infinitely many  $n \geq n_0$ ,*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \geq (\alpha - \epsilon)^{b+1} (1 - q)^{n(b+1)}.$$

Consequently, for any  $\alpha \in (0, 1)$ ,

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \leq (b + 1) \log \frac{1}{1 - q}.$$

*Proof.* Please see Appendix A.11. □

The next claim establishes that the proposed iLQF with PullUp algorithm results in a matching lower bound on the rate function for the system with asymmetric arrivals, and is therefore rate function optimal for the small buffer overflow event for this system.

**Theorem 2.6.** *For the system with asymmetric arrivals, the iLQF with PullUp algorithm has the property*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \geq (b+1) \log \frac{1}{1-q}.$$

*Proof.* Please see Appendix A.12. □

As a result of Theorems 2.5 and 2.6, the proposed algorithm (iLQF with PullUp) is rate-function optimal for the system with asymmetric arrivals.

### 2.6.2 Symmetric, Bursty, ON-OFF Arrivals

Let the arrival process for the system  $\Upsilon'_n$  be given by

$$A_i^{[n]}(t) = \begin{cases} L & \text{with probability } p, \\ 0 & \text{with probability } 1-p, \end{cases}$$

for some fixed constants  $p$  and  $L$ , with  $pL \in (0, 1)$ . We refer to this system as a system with symmetric, bursty, ON-OFF arrivals. Note that the channel process of this system is exactly as that of the system  $\Upsilon_n$  defined in Section 2.1.

Hereafter in this section, for ease of notation, we drop the explicit dependence of the variables on  $n$ . As before, we let  $\ell_t := \max_{1 \leq i \leq n} Q_i(t)$ .

**Theorem 2.7.** *For a system with symmetric, bursty, ON-OFF arrivals implementing any algorithm for assigning servers to queues, and for all  $b \geq 0$ ,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \leq \left\lceil \frac{b+1}{L} \right\rceil \log \frac{1}{1-q}.$$

*Proof.* The proof is similar to that of Theorem 2.1, and presented in Appendix A.13.  $\square$

**Lemma 2.11.** *Fix any  $\tilde{p} \in (p, 1/L)$ . Define  $\Theta_1 := nH(\tilde{p}|p)$  and  $\Theta_2 := 3n\tilde{p}(1-q)^{n\tilde{p}}$ . Then for the symmetric, bursty, ON-OFF arrivals system, for  $n$  large enough, and for all  $t$ ,*

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(t+1) > \max_{1 \leq i \leq n} Q_i(t)\right) \leq \exp(-\Theta_1) + L\Theta_2.$$

*Proof.* Please see Appendix A.14.  $\square$

**Lemma 2.12.** *For the symmetric, bursty, ON-OFF arrivals system implementing the  $iLQF$  with PullUp algorithm, there exists a constant  $k = k(L, p) = \left\lceil \frac{3}{1-pL} \right\rceil$  such that for all  $n$  large enough, for all  $m > 0$  and all  $T$ ,*

$$\mathbb{P}(\ell_{T+k} < m \mid \ell_T = m) \geq \frac{1}{2}.$$

*Proof.* Please see Appendix A.15.  $\square$

**Theorem 2.8.** *For a system with symmetric, bursty, ON-OFF arrivals implementing the  $iLQF$  with PullUp algorithm, for all  $b \geq 0$ , and for  $\tilde{p} = (1/L+p)/2$ ,*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i^{[n]}(0) > b\right) \geq \left\lceil \frac{b+1}{L} \right\rceil \min\left(\tilde{p} \log \frac{1}{1-q}, H(\tilde{p}|p)\right) > 0.$$

*Proof.* Please see Appendix A.16.  $\square$

### 2.6.3 Symmetric Arrivals with Bounded Support

Let the arrival process for the system  $\Upsilon'_n$  be given by

$$A_i^{[n]}(t) = \begin{cases} 0 & \text{with probability } p_0, \\ 1 & \text{with probability } p_1, \\ \vdots & \vdots \\ L & \text{with probability } p_L, \end{cases} \quad (2.6.3)$$

and  $\mathbb{P}(A_i^{[n]}(t) > L) = 0$ . We require that  $p_i \geq 0$  for all  $i$ ,  $p_L > 0$ ,  $\sum_{i=0}^L p_i = 1$  and  $\sum_{i=0}^L ip_i \in (0, 1)$  for stability for  $n$  large. We refer to this system as a system with symmetric arrivals with bounded support.

**Notation:**

Let  $r = [r_0, r_1, \dots, r_L]$  be a probability vector, that is,  $r_i \geq 0$  for all  $i$  and  $\sum_i r_i = 1$ . Let  $M_1(\Sigma)$  denote the probability simplex in  $\mathfrak{R}^{L+1}$ , that is, the set of all probability vectors in  $\mathfrak{R}^{L+1}$ .

Let  $\lambda_p := \sum_i ip_i < 1$  denote the expected number of arrivals to a queue in the system. Define

$$F_\epsilon := \left\{ r = [r_0, r_1, \dots, r_L] : r_k \geq 0 \text{ for all } k, \sum_{k=0}^L r_k = 1, \right. \\ \left. p_k = 0 \Rightarrow r_k = 0, \sum_{k=0}^L kr_k \leq \frac{1 + \lambda_p}{2}, \max_{0 \leq k \leq L} |r_k - p_k| \leq \epsilon \right\}.$$

For all  $\epsilon > 0$ , the set  $F_\epsilon$  is nonempty ( $\because p \in F_\epsilon$ ) and compact. There exists  $\epsilon_0 \in (0, p_L)$  such that for all  $\epsilon \leq \epsilon_0$ , the complement of the set  $F_\epsilon$  with respect to  $M_1(\Sigma)$  is contained in the closure of its interior w.r.t.  $M_1(\Sigma)$ . Since  $H(r|p) = 0$

if and only if  $r = p$ , and using results from [7], Section 2.1.1, it follows that

$$\zeta := \inf_{s \in M_1(\Sigma) \setminus F_{\epsilon_0}} H(s|p) > 0.$$

Define  $J := \min_{d \in F_{\epsilon_0}} d_L$ , where  $d_L$  denotes the  $L^{\text{th}}$  co-ordinate of  $d = [d_0, d_1, \dots, d_L]$ .

**Theorem 2.9.** *For the system with symmetric arrivals with bounded support, under the  $iLQF$  with pullup algorithm, for all  $b \geq 0$ ,*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \geq \left\lceil \frac{b+1}{L} \right\rceil \min \left( J \log \frac{1}{1-q}, \zeta \right) > 0.$$

*Proof.* Please see Appendix A.17. □

**Theorem 2.10.** *For a system with symmetric arrivals with bounded support, implementing any algorithm for assigning servers to queues, and for all  $b \geq 0$ ,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \leq \left\lceil \frac{b+1}{L} \right\rceil \log \frac{1}{1-q}.$$

*Proof.* Similar to that of Theorem 2.7. □

#### 2.6.4 Asymmetric Arrivals with Bounded Support

Let the arrival process for the system  $\Upsilon'_n$  be given by

$$A_i^{[n]}(t) = \begin{cases} 0 & \text{with probability } p_i^{(n)}(0), \\ 1 & \text{with probability } p_i^{(n)}(1), \\ \vdots & \vdots \\ L & \text{with probability } p_i^{(n)}(L), \end{cases} \quad (2.6.4)$$

and  $\mathbb{P}(A_i^{[n]}(t) > L) = 0$ . We require  $p_i^{(n)}(j) \geq 0$  for all  $i, j$ ,  $\sum_{j=0}^L p_i^{(n)}(j) = 1$  for all  $i$  and (for stability for  $n$  large)

$$\limsup_{n \rightarrow \infty} \max_{1 \leq i \leq n} \sum_{j=0}^L j p_i^{(n)}(j) \in (0, 1).$$

**Theorem 2.11.** *For the system under consideration, under the iLQF with pullup algorithm, for all  $b \geq 0$ ,*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) > 0.$$

*Proof.* Similar to that of Theorems 2.6 and 2.9. □

Thus, we see that the proposed iLQF with PullUp algorithm is robust to changes in the system model, and results in a strictly positive value of the rate function for the small buffer overflow event under a number of scenarios.

## Chapter 3

### Low-complexity Scheduling Algorithms

In Chapter 2, we looked at the problem of designing scheduling algorithms for a multi-user multi-channel (e.g., OFDM-based) wireless downlink system, where the grand goal is to design a scheduling rule for allocating the servers to the queues that, in addition to throughput-optimality, also guarantees small per-user queues in a large deviations sense. In [3], we have shown that a class of algorithms called iLQF (iterated Longest Queues First), under certain technical conditions, is rate function optimal for the small buffer overflow event, and the results are summarized in Chapter 2. However, the results regarding iLQF were derived assuming symmetric, i.i.d., ON-OFF traffic and i.i.d. ON-OFF channels. In particular, for more general arrival and channel processes, a number of fundamental questions were left unanswered, such as:

1. Are the algorithms in the iLQF class throughput-optimal for the system?
2. The well-known MaxWeight algorithm [30] is throughput-optimal for the system. What is its performance for the small buffer overflow problem?
3. The iLQF-class algorithms typically have a higher computational complexity than the MaxWeight algorithm. Can we design an algorithm with

lower complexity without compromising either throughput-optimality or small-queue performance?

In this chapter, we show that the answers to these questions are yes, poor, and yes respectively. The following is a summary of our main results in this chapter:

- We show that a generalization of the iLQF-class rules, namely iHLQF, is throughput-optimal for very general arrival and channel process models (Section 3.1).
- We show that the classical MaxWeight rule is very poor at keeping the per-user queues small (Section 3.2). Formally, we show that this rule results in a zero rate function for the small buffer overflow event defined in Section 2.1.
- We propose a new scheduling algorithm called the Server-Side Greedy (SSG) service rule which is an iterative version of the MaxWeight rule, where the queue-lengths are updated after each server (OFDM sub-channel) finishes its service. We show that this rule is throughput-optimal under general arrival and channel processes, results in a strictly positive rate function for the small buffer overflow event (implying small per-user queues), and has complexity comparable to that of the MaxWeight rule, and much less than the iLQF-class rules (Section 3.3).

An important design insight that emerges from the above results is the following: for throughput-optimality, the MaxWeight algorithm argues that the scheduling algorithm should maximize the sum of channel-rate-weighted queue lengths at each timeslot. However, from a small-queue performance viewpoint, our results indicate that as long as we are “close” to the maximum weighted sum, the scheduling algorithm’s objective should shift to equalizing the queues, and that this allocation of channel resources should proceed in an iterative manner.

We consider the same multi-user multi-channel discrete-time queuing system defined in Section 2.1, but make certain assumptions on the arrival and channel processes as follows:

**Assumption 3.1.** *(Motivated by [10])*

***The channel state process:***

1. Let  $\mathcal{J}$  denote the set of possible channel states. The channel state process has a stationary distribution  $\pi = [\pi_i]_{i \in \mathcal{J}}$ , and  $\pi_i > 0$  for all  $i \in \mathcal{J}$ .
2. Let  $s[m]$  denote the channel state in timeslot  $m$ . Given  $\epsilon > 0$  there exists a positive integer  $M_0$  such that for all  $M \geq M_0$ , all  $i \in \mathcal{J}$ , and all  $k$ , we have

$$\mathbb{E} \left[ \left| \pi_i - \frac{1}{M} \sum_{m=k}^{k+M-1} \mathbf{1}_{s[m]=i} \right| \right] < \epsilon.$$

3. There exists  $\hat{\mu} \in \mathfrak{R}_+$  such that  $X_{ij}(t) \leq \hat{\mu}$  for all  $i, j, t$ .

***The arrival process:***

1. The arrivals to each queue  $Q_i$  form a stationary process, with mean  $\lambda_i := \mathbb{E}[A_i(1)]$ .
2. The given set of arrival rates lies inside the throughput region of the system, i.e., there exists a static service split rule that can stabilize the system under the given arrival process.
3. Given  $\epsilon > 0$  there exists a positive integer  $M_1$  such that for all  $M \geq M_1$ , and for all  $k$ , we have

$$\left[ \left| \lambda_i - \frac{1}{M} \sum_{m=k}^{k+M-1} A_i(m) \right| \right] < \epsilon, \quad \forall i.$$

4. The arrival process satisfies, for all  $i$ ,

$$\lim_{M \rightarrow \infty} M^2 \mathbb{P}(A_i(1) > M) = 0. \quad \diamond$$

Our first goal is to identify policies other than MaxWeight-type policies which are throughput-optimal under Assumption 3.1. The reason for identifying classes of throughput-optimal policies other than the MaxWeight class is to significantly improve the performance of the system. It is difficult to analyze the large-deviations performance under the very general conditions in Assumption 3.1. Instead, we study the performance of different algorithms under the following more restrictive set of assumptions:

**Assumption 3.2.** *The number of packet arrivals to queue  $Q_i$  in timeslot  $t$  is the random variable  $A_i(t)$ , where*

$$A_i(t) = \begin{cases} \bar{K} & \text{with probability } p, \\ 0 & \text{with probability } 1 - p, \end{cases}$$

where  $\bar{K} \geq 1$  is an integer with  $p\bar{K} \in (0, 1)$ . In timeslot  $t$ , the server  $S_j$  can potentially serve  $X_{ij}(t)$  packets from  $Q_i$ , where  $X_{ij}(t)$  are modeled as Bernoulli random variables with

$$X_{ij}(t) = \begin{cases} 1 & \text{with probability } q, \\ 0 & \text{with probability } 1 - q, \end{cases}$$

with  $q \in (0, 1)$ . All the random variables  $A_i(t)$  and  $X_{jk}(s)$  are assumed to be mutually independent for all possible values of the involved parameters.  $\diamond$

As in Chapter 2, our goal is to design policies that under Assumption 3.2 and for every integer  $b \geq 0$ , result in a strictly positive value of the rate-function for the small-buffer overflow event,

$$I(b) := \liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right).$$

Further, the complexity of the policy must be  $\mathcal{O}(n^2)$  computations per timeslot. The reason we choose this benchmark is that, as we later show, the MaxWeight algorithm has a complexity  $\Omega(n^2)$  computations per timeslot.

To summarize, our objective is to design an algorithm that is throughput-optimal under Assumption 3.1, results in a strictly positive value of the rate function under Assumption 3.2, and has a complexity of  $\mathcal{O}(n^2)$  computations per timeslot. We close this introduction with a result regarding the system stability under Assumption 3.2.

**Lemma 3.1.** *Under Assumption 3.2, if  $p\bar{K} > 1$ , then the system is unstable under any scheduling algorithm. If  $p\bar{K} < 1$ , then there exists a constant  $n_0 = n_0(p, \bar{K}, q)$  such that for all  $n \geq n_0$ , the system is stable under some algorithm.*

*Proof.* Please see Appendix B.1. □

Note that the stability of the system with  $p\bar{K} < 1$  is immediate from Theorem 2.8, which shows that the iLQF with PullUp algorithm results in a strictly positive rate-function for the small-buffer overflow event. The above result establishes that this condition is also necessary for stability.

### 3.1 iHLQF Analysis

In this section, we present a class of scheduling rules called iHLQF. We show that any algorithm in this class is throughput-optimal (Theorem 3.1), and relate the iLQF with PullUp algorithm in Chapter 2, Section 2.5 to the iHLQF-class.

We consider a class of algorithms called iHLQF (iterated Heaviest matching with Longest Queues First), which is a generalization of the iLQF (iterated Longest Queues First) algorithms presented in Chapter 2, Section 2.4. The iLQF-class of algorithms is defined for systems with ON-OFF channels. A particular algorithm in the iLQF-class, called iLQF with PullUp, was shown to be rate function optimal for the small buffer overflow event under Assumption 3.2 with  $\bar{K} = 1$  ([3], Thm. 4, also Theorem 2.4, Chapter 2), and was shown to have a positive rate function for all  $\bar{K} > 0$  ([3], Corollary 2, also Theorem 2.8, Chapter 2). Here we present its generalization (namely, iHLQF) to multi-rate channels (i.e.,  $X_{ij}(t)$  can take value other than 0 and 1). In every timeslot, the iHLQF rule proceeds in multiple rounds of server allocation as

explained below.

**Definition 3.1** (The iHLQF (iterated Heaviest matching with Longest Queues First) rule). *The iHLQF rule allocates servers to queues in timeslot  $t$  according to the following procedure.*

**Input:**

1. The queue-lengths  $Q_i(t-1)$  for  $1 \leq i \leq n$ .
2. The arrival vector  $A_i(t)$  for  $1 \leq i \leq n$ .
3. The channel rates  $X_{ij}(t)$  for  $1 \leq i, j \leq n$ .

**Steps:**

1. Update the queue-length vector to account for the arrivals. Compute  $Q_i^{(0)}(t) = Q_i(t-1) + A_i(t)$  for all  $i$ . Initialize  $k = 1$  and  $Y_{ij}(t) = 0$  for  $1 \leq i, j \leq n$ .
2. For all  $j \in \{1, 2, \dots, n\}$ , define

$$\mathcal{T}_j := \arg \max_{1 \leq i \leq n} Q_i^{(0)}(t) X_{ij}(t),$$

$$\text{and } c_j := \min \left[ \arg \max_{m \in \mathcal{T}_j} X_{mj}(t) \right].$$

For any server  $S_j$  and all  $i$ , if  $X_{ij}(t) < X_{c_j j}(t)$ , then redefine  $X_{ij}(t) = 0$ , and use this updated value of  $X_{ij}(t)$  throughout the rest of the timeslot  $t$ . Let  $L$  denote the length of the longest queue(s) immediately after arrivals. Throughout the description of this algorithm, let  $\mathcal{V} \subseteq \mathcal{S}$  denote the set of unallocated servers.

3. In round  $k$ , define a bipartite graph  $G_k(\mathcal{U}_k \cup \mathcal{V}_k, \mathcal{E}_k)$ , where the set of nodes  $\mathcal{U}_k$  represents the set of queues of length  $L$  (i.e.  $Q_i^{(k-1)}(t) = L$ ), the set of nodes  $\mathcal{V}_k$  represents the servers in  $\mathcal{V}$ , and  $\mathcal{E}_k$  is the set of weighted edges. The edge between nodes representing queue  $Q_i$  and server  $S_j$  has a weight equal to  $X_{ij}(t)$ . Find a maximum weight matching  $\mathcal{M}_k$  in the graph  $G_k$ , breaking ties arbitrarily. Allocate the servers to the queues according to the matching  $\mathcal{M}_k$ , update the queue-lengths and remove the used servers from further consideration. In particular, if  $Q_i$  is allocated  $S_j$ , then define  $Q_i^{(k)}(t) = (Q_i^{(k-1)}(t) - X_{ij}(t))^+$ , remove the server  $S_j$  from  $\mathcal{V}$ , and define  $Y_{ij}(t) = 1$ .
4. If  $\mathcal{V} = \emptyset$ , then define  $k_0 := k$  and stop. Else, decrement  $L$  by 1. If  $L = 0$ , then define  $k_0 := k$  and stop. Else, increment  $k$  by 1, and go to Step 3.

**Output:**

1. The server allocations,  $Y_{ij}(t)$  for  $1 \leq i, j \leq n$ .
2. The queue-lengths  $Q_i(t) := Q_i^{(k_0)}(t)$  for  $1 \leq i \leq n$ . ◇

Here is a description of the algorithm in words: in every timeslot, have multiple rounds of server allocation. In each round, choose the heaviest (edge-weight) matching between the set of *longest* queues and available servers, breaking ties between multiple heaviest matchings arbitrarily. Here, the weight of an edge is the corresponding channel rate. Allocate servers to queues according to the matching, *update the queue-lengths*, remove the allocated servers

from further consideration, and proceed to the next round. When choosing the heaviest matching, the iHLQF rule allocates a server to a queue only if its channel to that queue has a high enough rate, as evident from the Step 2 in the definition.

Note that iHLQF is a class of rules and not a single rule, as a result of the arbitrary tie-breaking between largest weight matchings. We first establish a crucial property of the iHLQF-class rules that is useful in proving their throughput-optimality. In words, this property says that in any given timeslot, the weight of the schedule chosen by any iHLQF-class rule is at most an additive constant away from that chosen under the throughput-optimal MaxWeight rule under the same initial queue-lengths, same arrivals to each queue and the same channel realizations.

Fix any timeslot  $t$ . At the beginning of the timeslot, let the state of the system be denoted by the queue-lengths  $Q_i(t-1)$ , channel realizations  $X_{ij}(t)$  and arrivals  $A_i(t)$ . Let

$$\sum_{i=1}^n (Q_i(t-1) + A_i(t)) X_{ij}(t) Y_{ij}(t)$$

be called the weight contributed by server  $S_j$  to a schedule. Note that at most one term in the above summation is nonzero, since  $Y_{ij}(t) = 1$  for at most one value of  $i$ . Let the weight of a schedule  $R$  in timeslot  $t$  (for the given state) be defined as

$$W^R(t) := \sum_{j=1}^n \sum_{i=1}^n (Q_i(t-1) + A_i(t)) X_{ij}(t) Y_{ij}(t),$$

that is, the sum of the weight-contributions of the individual servers. The proof of the next lemma proceeds by showing that conditioned the same initial state, the contribution of any server  $S_j$  to the weight of the MaxWeight schedule is at most an additive constant more than that to the schedule of any (fixed) iHLQF-class rule.

**Lemma 3.2.** *Fix any iHLQF-class rule  $\Lambda$ . Fix any timeslot  $t$  and the state of the system. Let the length of the queue  $Q_i$  immediately after arrivals be  $\ell_i := Q_i(t-1) + A_i(t)$ . Let  $c_j$  be as defined in step 2 in the definition of the iHLQF-class rules (Definition 3.1). Then,  $W^{iHLQF}(t) \geq W^{MW}(t) - n^2 \hat{\mu}^2$ .*

*Proof.* Please see Appendix B.2. □

**Theorem 3.1** (Throughput-optimality of iHLQF). *Under Assumption 3.1 on the arrival and channel processes, any iHLQF-class rule makes the system stable in the mean, i.e.,*

$$\limsup_{p \rightarrow \infty} \frac{1}{p} \sum_{k=0}^{p-1} \mathbb{E} \left[ \sqrt{\sum_{i=1}^n Q_i^2(k)} \right] < \infty.$$

*In addition, if the arrival and channel state processes are such that the iHLQF rule makes the queuing system an aperiodic Markov chain with a single communicating class, then the stability in the mean implies that the Markov chain is positive recurrent [20].*

*Proof.* In view of Lemma 3.2, this proof is identical to that of Theorem 3.5 presented in Section 3.3 and has been omitted to avoid repetition. □

Next, we establish a result regarding the computational complexity of iHLQF-class algorithms.

**Lemma 3.3.** *(Complexity of iHLQF) There exists an algorithm in the iHLQF class that can be implemented in  $\mathcal{O}(n^4)$  computations per timeslot.*

*Proof.* Please see Appendix B.3. □

We now turn our attention to the iLQF with PullUp rule, introduced in [3] (also see Definition 2.7, Chapter 2). This rule is essentially an iHLQF-class rule for the system under Assumption 3.2. The iLQF with PullUp rule employs a particular form of tie-breaking between the heaviest matchings, and in its original form, it terminates after the round when it cannot find a matching that serves all the queues under consideration (queues of length  $L$ ). This stopping rule ensures that the complexity of the rule is limited to  $\mathcal{O}(n^4)$  computations per timeslot. If we instead allow the iLQF with PullUp rule to terminate in the same way as an iHLQF-class rule (namely, when no more servers are left or when all the nonempty queues have been considered for allocation), then it retains its rate-function optimality for the system under Assumption 3.2 with  $\bar{K} = 1$ , yields a strictly positive rate function for all  $\bar{K} \geq 1$ , and is also throughput-optimal for the system under Assumption 3.1 with Bernoulli (0-1) channels. We refer to this rule as the Modified iLQF with PullUp rule. The following theorem formally summarizes these properties.

**Theorem 3.2** (Properties of Modified iLQF with PullUp). *The Modified iLQF with PullUp belongs to the iHLQF class of rules, and the conclusions of The-*

orem 3.1 apply. The Modified iLQF with PullUp rule is rate function optimal for the system under Assumption 3.2 with  $\bar{K} = 1$ , and results in

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = (b + 1) \log \frac{1}{1 - q},$$

for all integers  $b \geq 0$ . Further, it yields a strictly positive rate function under Assumption 3.2 for all  $\bar{K} \geq 1$ , and can be implemented in  $\mathcal{O}(n^5)$  computations per timeslot.

*Proof.* Omitted to avoid repetition. For the proof of rate function optimality, we refer the reader to the proof of Theorem 2.4, Chapter 2 (or [3]). Lemma 2.6 establishes the computational complexity result.  $\square$

Thus, the answer to question 1 at the beginning of this chapter is “yes, the iLQF rules (and their generalization, iHLQF, for multi-rate channels) are throughput-optimal for the system.”

## 3.2 The MaxWeight Rule

In this section, we show that the classic MaxWeight scheduling rule yields a zero rate function for the small buffer overflow event (Theorem 3.3), and in fact yields no decay in the probability of the small buffer overflow event as the system size increases (Theorem 3.4). We then establish a lower bound (computations per timeslot) on the complexity of the MaxWeight rule (Lemma 3.4).

The classic MaxWeight rule [30] results in the following service allocation rule for our system, with a particular tie-breaking rule for the sake of concreteness:

**Definition 3.2** (The MaxWeight Rule, [30]). *In every timeslot, each server independently picks a queue that maximizes the product of queue-length and channel rate, breaking ties in favor of the smallest queue-index.*  $\diamond$

This service allocation rule is throughput-optimal for the system. But as the next theorem shows, it yields a zero rate function for the small buffer overflow event, implying a very poor small-queue performance.

**Theorem 3.3** (MaxWeight gives zero rate function). *Under Assumption 3.2 with  $\bar{K} = 1$ , with the MaxWeight rule for allocating servers to queues, and for any given integer  $b \geq 0$ , there exists an integer  $M_0$  such that for all  $n \geq M_0$ , we have*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \frac{1}{2}.$$

*Consequently, for every integer  $b \geq 0$ , we have*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = 0.$$

The main idea behind the proof is to show that under the MaxWeight rule, the overflow event has at least a constant probability even for  $n$  large.

*Proof.* Please see Appendix B.4.  $\square$

The result of Theorem 3.3 can be strengthened to the following:

**Theorem 3.4.** *Consider any function  $f : \mathfrak{R}_+ \rightarrow \mathfrak{R}_+ \setminus \{0\}$  such that  $\lim_{x \rightarrow \infty} f(x) = \infty$ . Then, under Assumption 3.2, with the MaxWeight rule for allocating the servers to the queues, and for any fixed integer  $b \geq 0$ ,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{f(n)} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = 0.$$

*Proof.* The result for the case  $\bar{K} = 1$  follows from the proof of Theorem 3.3, which shows that under Assumption 3.2 with  $\bar{K} = 1$ , the MaxWeight rule results in at least a constant probability for the small buffer overflow event, for  $n$  large enough. The proof for the case  $\bar{K} > 1$  is almost identical.  $\square$

Thus, the answer to question 2 at the beginning of this chapter is “the MaxWeight algorithm is very inefficient at keeping the per-user queues small.” The main reason behind these negative results is that the MaxWeight rule potentially assigns all the available servers to serve *the* longest queue, essentially treating a slightly shorter queue as if it were empty. When a large number of servers are available, this results in draining the longest queue(s) much more than is warranted by good load-balancing, and also leads to the following situation: when the MaxWeight rule runs into a state when a significant fraction of the queues is long (note that such a state is reached infinitely often, almost surely, because the system is positive recurrent under MaxWeight), then it is very difficult to leave this state “quickly.” Therefore, the MaxWeight rule is not effective in keeping the queues *really* small.

**Lemma 3.4** (Complexity of MaxWeight). *Under Assumption 3.2, implementing the MaxWeight rule requires  $\Omega(n^2)$  computations per timeslot.*

*Proof.* Please see Appendix B.5. □

Note that Lemma 3.4 holds under much weaker assumptions on the channel process. This motivates us to design a scheduling rule that, in addition to throughput-optimality, also guarantees a good delay performance, and has a computational complexity comparable to that of the MaxWeight rule.

### 3.3 The SSG Scheduling Rule

In this section, we propose the Server-Side Greedy (SSG) service rule for the problem. This service rule can be thought of as a recursive version of the MaxWeight rule, where the queue-lengths are updated after each server finishes its service. We show that the SSG rule is throughput-optimal for the system (Theorem 3.5). Under Assumption 3.2, it results in a strictly positive value of the rate function,  $\alpha(b)$  for every integer  $b \geq 0$  (Theorems 3.6, 3.7). It can be implemented in  $\mathcal{O}(n^2)$  computations per timeslot (Theorem 3.8).

This (SSG) rule proceeds in multiple rounds of service allocation in every timeslot, as explained below.

**Definition 3.3** (The SSG Rule). *The Server-Side Greedy (SSG) rule allocates servers to queues in timeslot  $t$  according to the following procedure.*

***Input:***

1. The queue-lengths  $Q_i(t-1)$  for  $1 \leq i \leq n$ .
2. The arrival vector  $A_i(t)$  for  $1 \leq i \leq n$ .
3. The channel rates  $X_{ij}(t)$  for  $1 \leq i, j \leq n$ .

**Steps:**

1. Update the queue-length vector to account for the arrivals, i.e., compute  $Q_i^{(0)}(t)$  for all  $i$ . Initialize  $k = 1$  and  $Y_{ij}(t) = 0$  for  $1 \leq i, j \leq n$ .
2. In the  $k^{\text{th}}$  round of service allocation, search for the queue-index

$$w \in \arg \max_{1 \leq i \leq n} Q_i^{(k-1)}(t) X_{ik}(t),$$

breaking ties in favor of the smaller queue-index. Allocate the server  $S_k$  to serve  $Q_w$  thus found, i.e., define  $Y_{wk}(t) = 1$  and  $Y_{ik}(t) = 0$  for all  $i \neq w$ . Update the length of  $Q_w$  so that

$$Q_w^{(k)}(t) = (Q_w^{(k-1)}(t) - X_{wk}(t))^+,$$

and the lengths of all other queues are left unchanged, i.e.,  $Q_i^{(k)}(t) = Q_i^{(k-1)}(t)$  for all  $i \neq w$ .

3. If  $k = n$ , then stop. Else, increment  $k$  by 1, go to step 2.

**Output:**

1. The server allocations,  $Y_{ij}(t)$  for  $1 \leq i, j \leq n$ .

2. The queue-lengths  $Q_i(t) := Q_i^{(n)}(t)$  for  $1 \leq i \leq n$ .  $\diamond$

Unlike the MaxWeight rule, the SSG rule updates queue-lengths after each server finishes its service. An example of the SSG rule is shown in Figure 3.1. Now we analyze the SSG service rule in detail. Our first aim is to

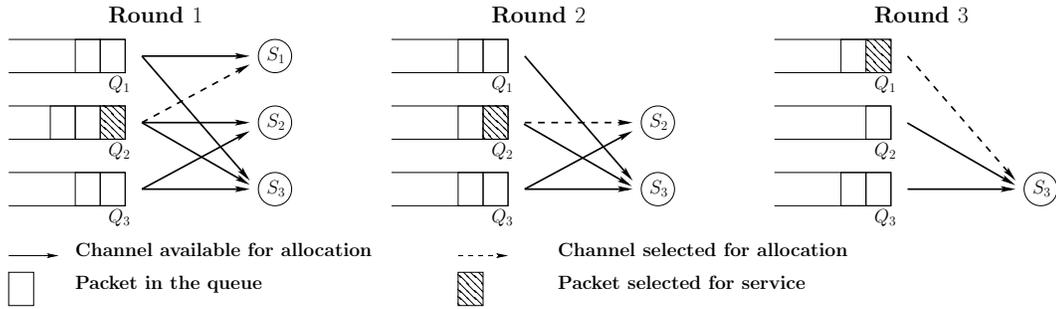


Figure 3.1: An example of the SSG rule

establish the throughput-optimality of the SSG rule. We first establish that in every timeslot, the weight of the service schedule selected by the SSG rule is at most an additive constant away from the maximum possible. That is, under the same queue-lengths at the beginning of the timeslot, the same channel realizations and the same arrivals to the queues, the sum of the channel-rate-weighted queue-lengths selected for service under the MaxWeight rule is at most an additive constant larger than that under the SSG rule.

**Lemma 3.5.** *Fix any timeslot  $t$ , and let the queue-lengths immediately after arrivals in that timeslot be  $l_i$ . Then,  $W^{SSG}(t) \geq W^{MW}(t) - n^2 \hat{\mu}^2$ .*

*Proof.* Please see Appendix B.6.  $\square$

**Theorem 3.5** (Throughput-optimality of SSG). *Under Assumption 3.1 on the arrival and channel processes, the SSG rule makes the system stable in the mean, i.e.,*

$$\limsup_{p \rightarrow \infty} \frac{1}{p} \sum_{k=0}^{p-1} \mathbb{E} \left[ \sqrt{\sum_{i=1}^n Q_i^2(k)} \right] < \infty.$$

*In addition, if the arrival and channel state processes are such that the SSG rule makes the queuing system an aperiodic Markov chain with a single communicating class, then the stability in the mean implies that the Markov chain is positive recurrent [20].*

*Proof.* Please see Appendix B.7. □

**Remark 3.1.** *Lemma 3.5 and Theorem 3.5 hold even if the SSG rule chooses an arbitrary tie-breaking rule in Step 2.*

Next, we show that if two queuing systems have sample-path coupled arrivals and channels, both implement the SSG rule, and at the end of a timeslot, one system has queues that are respectively longer than the corresponding queues in the second system, then this property continues to hold for all the future timeslots.

**Lemma 3.6** (Sample-path dominance). *Under Assumption 3.2, consider two queuing systems  $\underline{Q}$  and  $\underline{R}$  with queues  $\mathcal{Q} = [Q_1, Q_2, \dots, Q_n]$  and  $\mathcal{R} = [R_1, R_2, \dots, R_n]$ , such that at the end of some timeslot  $t$ , we have  $Q_i(t) \leq R_i(t)$  for all  $i$ . Both the systems have the same arrivals and channel realizations for all times, and in particular for the timeslot  $t + 1$ . Both implement the SSG rule. Then,  $Q_i(t + 1) \leq R_i(t + 1) \forall i$ .*

*Proof.* Please see Appendix B.8. □

As in the case of the proof of Theorem 2.4 in Chapter 2, this sample-path property is the key to obtaining rate function positivity results. The next technical lemma provides a sufficient condition for *all* the longest queues to be served by the SSG rule.

**Lemma 3.7.** *Under Assumption 3.2, let the set of longest queues after arrivals be of cardinality  $k$ . If in that timeslot, each one of the longest queues is connected to at least  $k$  servers, then all the longest queues are served at least once under SSG.*

*Proof.* Let  $\{Q_{i_1}, Q_{i_2}, \dots, Q_{i_k}\}$  be the set of longest queues. If  $Q_{i_j}$  is connected to server  $S_m$ , then it is a longest queue connected to  $S_m$ . Since  $Q_{i_j}$  is connected to at least  $k$  servers, there are only  $k - 1$  other longest queues in the system, and the queue-lengths are updated after service,  $Q_{i_j}$  is served by at least one server. □

We now show that under the SSG rule, the probability that the max. queue in the system increases in a given timeslot is extremely small for  $n$  large.

**Lemma 3.8.** *Let Assumption 3.2 hold with  $\bar{K} = 1$ . Fix any  $p' \in (p, 1)$  and  $\delta \in (0, \frac{q(1-p')}{2-q})$ . In particular, let  $p' = (1 + p)/2$  and  $\delta = \frac{q(1-p')}{2}$ . Then, under the SSG rule, for  $n$  large enough, and for any given  $t$ ,*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(t+1) > \max_{1 \leq i \leq n} Q_i(t) \right) \leq n(1-q)^{n\delta} + n\delta \exp \left( \frac{-2n\delta H(\frac{q}{2}|q)}{q} \right) + e^{-nH(p'|p)}.$$

*Proof.* Please see Appendix B.9. □

Next, we establish that under the SSG rule and for  $n$  large, the max. queue-length in the system decreases in a constant number of timeslots with at least a constant probability.

**Lemma 3.9.** *Under Assumption 3.2 with  $\bar{K} = 1$ , there exists a constant integer  $k$ , independent of  $n$ , such that for all  $n$  large enough, and for all  $t$ ,*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(t+k) < \max_{1 \leq i \leq n} Q_i(t) \mid \max_{1 \leq i \leq n} Q_i(t) > 0 \right) \geq \frac{1}{2}.$$

*Proof.* The intuition behind the proof is that in any given timeslot, there are approximately  $np$  arrivals to the system, and because a given server can potentially serve any one of approximately  $nq$  of the queues (with preference to the longer queues), the system has a service capacity for almost  $n$  packets. Thus, there is a net “drain” in the number of packets in the system. For a formal proof, please see Appendix B.10. □

As a consequence of Lemmas 3.8 and 3.9, the maximum queue-length in the system,  $Q_{max}(t)$  has the following behavior: over a constant number of timeslots, it increases by a finite amount with very low probability, and decreases with at least a constant probability. Thus, it is reasonable to expect that the stationary distribution of  $Q_{max}(t)$  is strongly concentrated around 0. Indeed, this is the essence of our next claim.

**Theorem 3.6** (Positive rate function under SSG). *Let Assumption 3.2 hold with  $\bar{K} = 1$ . Fix any  $p' \in (p, 1)$  and  $\delta \in (0, \frac{q(1-p')}{2-q})$ . In particular, let  $p' = (1+p)/2$  and  $\delta = \frac{q(1-p')}{2}$ . Fix any constant  $b \geq 0$ . If the system uses the SSG rule for allocating servers to queues, then*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq (b+1) \min \left\{ H(p'|p), \delta \log \frac{1}{1-q}, \frac{2\delta H(\frac{q}{2}|q)}{q} \right\} > 0.$$

*Proof.* The proof is similar to that of Theorem 2.4 in Chapter 2. In particular, under the SSG rule, the system has the sample-path dominance property (Lemma 3.6), an exponentially decaying probability of  $Q_{max}(t)$  increasing in a timeslot (Lemma 3.8), and a constant probability of  $Q_{max}(t)$  decreasing in a constant number of timeslots (Lemma 3.9). We omit the details.  $\square$

Next, we show that the SSG rule returns a positive rate function for the small buffer overflow event under a bursty arrival process (Assumption 3.2), for any fixed integer  $\bar{K} \geq 1$ .

**Theorem 3.7.** *Let Assumption 3.2 hold. Fix any  $p' \in (p, 1/\bar{K})$  and  $\delta \in (0, \frac{q(1-p'\bar{K})}{K(2-q)})$ . In particular, let  $p' = (p + 1/\bar{K})/2$  and  $\delta = \frac{q(1-p'\bar{K})}{2K}$ . Fix any constant  $b \geq 0$ . If the system uses the SSG rule for allocating servers to queues, then*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \left\lceil \frac{b+1}{\bar{K}} \right\rceil \min \left\{ H(p'|p), \delta \log \frac{1}{1-q}, \frac{2\delta H(\frac{q}{2}|q)}{q} \right\} > 0.$$

*Proof.* The proof is very similar to the proof of Theorem 3.6 and has been omitted.  $\square$

An immediate strengthening of the above result is obtained by maximizing the RHS with respect to  $p'$  and  $\delta$  over the appropriate ranges. We now turn our attention to the computational complexity of implementing the SSG rule.

**Theorem 3.8** (Complexity of the SSG rule). *The SSG rule can be implemented in  $\mathcal{O}(n^2)$  computations per timeslot.*

*Proof.* Please see Appendix B.11.  $\square$

Thus, we have designed a scheduling algorithm (the SSG rule) that is throughput-optimal, yields a positive rate function for the small buffer overflow event, and has a computational complexity  $\mathcal{O}(n^2)$ , which is no larger than the MaxWeight rule. This answers question 3 at the beginning of the chapter in the affirmative. In view of these results, the new intuition that emerges from this work is that we should not allocate service to maximize the channel-weighted sum of queue-lengths. We should allocate resources in an iterative fashion, taking into account the effects of prior allocations. This results in good performance (small per-user queues) in addition to throughput-optimality.

### 3.4 Simulation Results

We compare the performance of the proposed SSG rule with the MaxWeight rule and the Modified iLQF with PullUp rule (Section 3.1). In the first set of simulations, we consider a system with bursty arrivals as per Assumption 3.2 with  $\bar{K} = 10$  and  $p = 0.095$ , i.e. a system with 95% of the maximum symmetric load. The channel ON probability is set to  $q = 0.75$ . We run the simulations for  $10^6$  timeslots, vary the number ( $n$ ) of queues and servers in the system, and study the empirical probability of buffer overflow and the empirical delay distribution of packets. The results are summarized in Figures 3.2 and 3.3. It can be seen that depending upon the system size, the MaxWeight algorithm needs about 3 to 7 times as much buffer as SSG. Further, the performance of the MaxWeight algorithm actually *degrades* with the system size, while that of the SSG improves. Similar conclusions hold for the per-packet delay under the two algorithms.

In the second set of simulations (Figure 3.4), we compare the performance of the SSG algorithm against the Modified iLQF with PullUp algorithm. We analyze a system with asymmetric arrival rates to the queues. Of the  $n = 20$  queues, we choose three queues to receive much higher mean loads ( $L$ ) compared to the others. In our simulations, queues  $Q_{11}, Q_{15}$  and  $Q_{19}$  receive, in every timeslot, a random number of packets that is uniformly distributed in  $[0, 2L]$ , while the other queues each receive a packet with probability 0.12, all independently of each other. We set the channel ON probability to  $q = 0.4$  to ensure that the system is stable but heavily loaded (about 95.7%

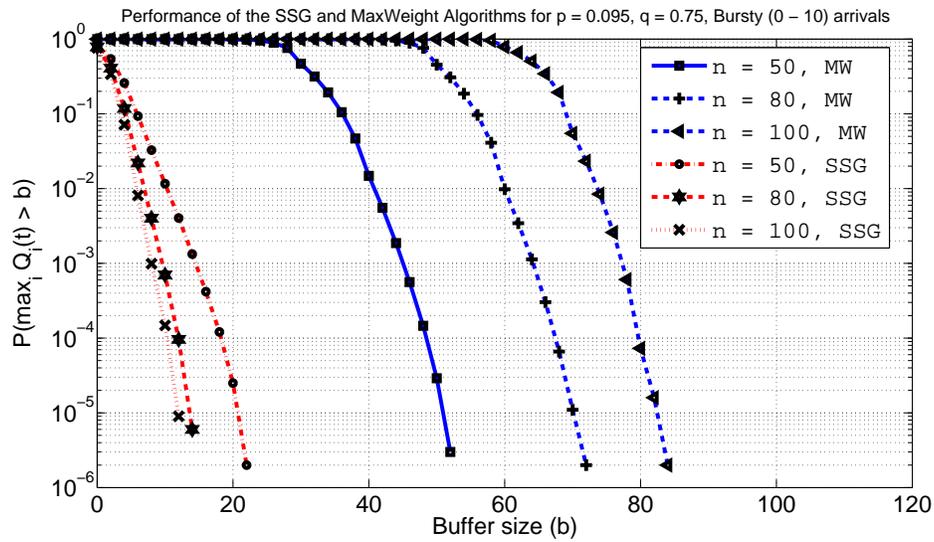


Figure 3.2: SSG v/s MaxWeight: Bursty load, buffer overflow probabilities

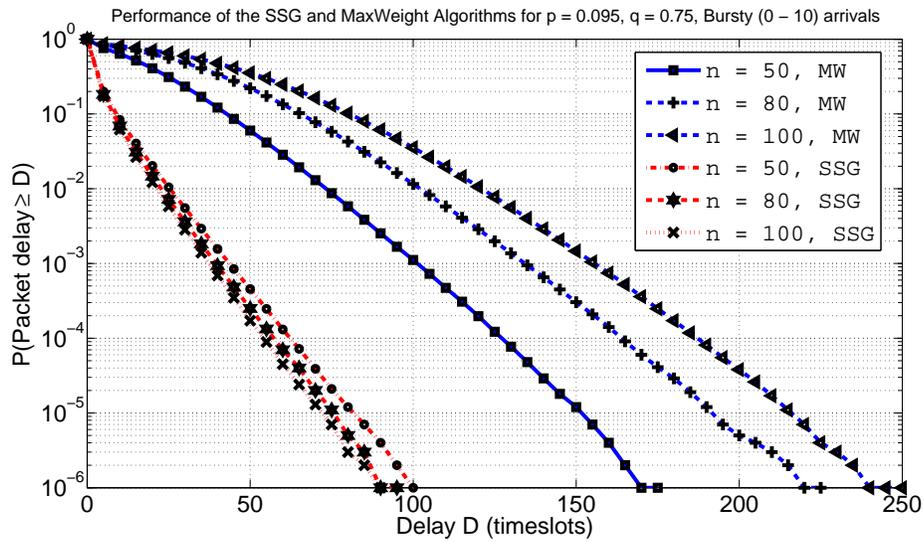


Figure 3.3: SSG v/s MaxWeight: Bursty load, packet delay profiles

for  $L = 5$ ). We run the simulations for  $10^6$  timeslots. As can be seen from the plots (Figures 3.4 and 3.5), the proposed SSG algorithm and the Modified iLQF with PullUp algorithm give very similar performance, both for the small buffer overflow probabilities and delay profiles. In fact, we can hardly distinguish the two curves in each pair. Even if the Y-axis scale is changed from logarithmic to linear, the two algorithms result in almost overlapping curves.

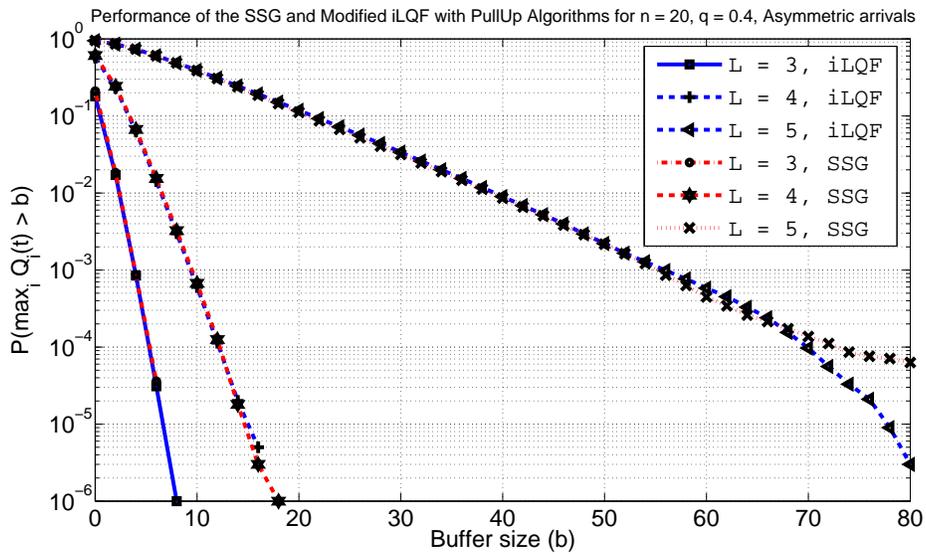


Figure 3.4: SSG v/s Modified iLQF with PullUp: Asymmetric arrivals, buffer overflow probabilities

The third set of simulations has the same set up as the first set, but we modify the MaxWeight rule as follows: instead of choosing the longest queue with the smallest index, each server now selects a longest queue that has received the minimum service so far (from the previous server allocation rounds), in the current timeslot. The servers are allocated sequentially, from  $S_1$  to  $S_n$ .

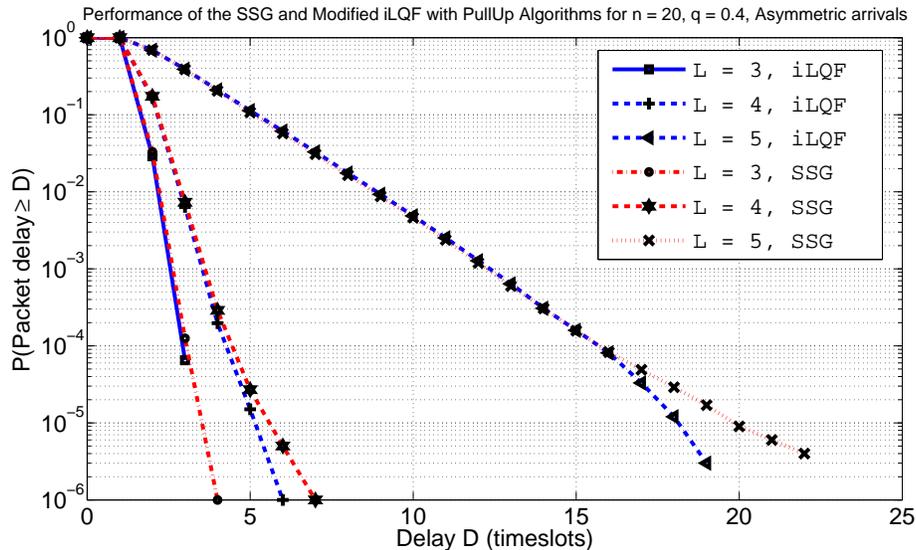


Figure 3.5: SSG v/s Modified iLQF with PullUp: Asymmetric arrivals, packet delay profiles

We compare the buffer and delay performance of this Modified MaxWeight algorithm and the SSG algorithm. The results are summarized in Figures 3.6 and 3.7. It can be seen that even under this implementation, the MaxWeight algorithm continues to perform much worse than the SSG algorithm.

In the fourth set of simulations, we considered a system with  $n = 20$  queues and servers. We set the channel ON probability to  $q = 0.5$ , and run the simulations for  $5 \times 10^5$  timeslots. The arrival process is according to Assumption 3.2, with  $\bar{K} = 4$ . We vary the probability  $p$  of packet arrivals, and compare the performance of the SSG and the Modified iLQF with PullUp algorithms. At  $p = 0.24$ , the system operates at 96% of the maximum stable load. As can be seen from Figure 3.8, the two algorithms result in almost

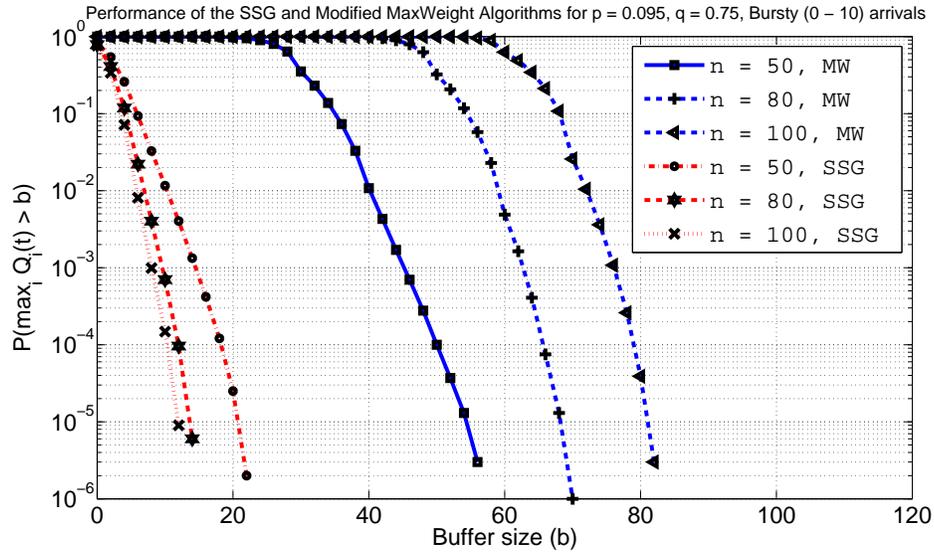


Figure 3.6: SSG v/s Modified MaxWeight: Bursty load, buffer overflow probabilities

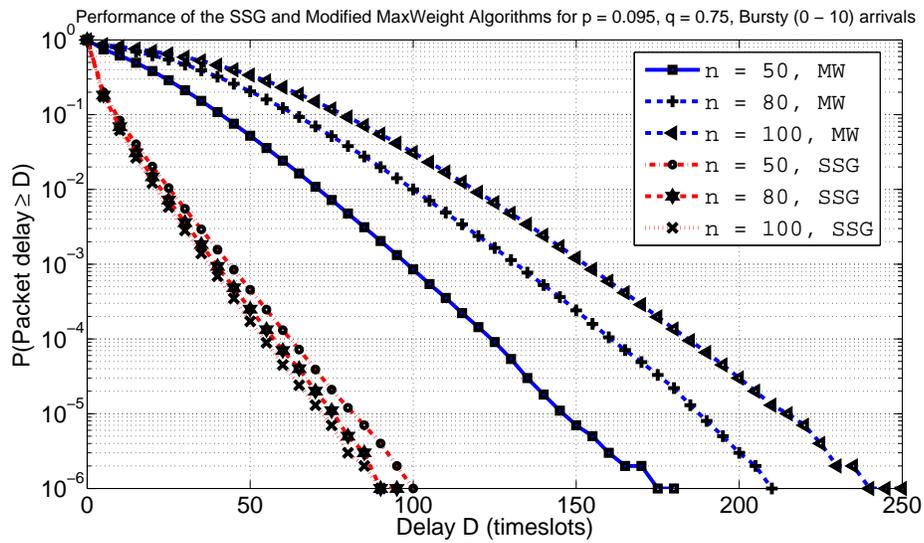


Figure 3.7: SSG v/s Modified MaxWeight: Bursty load, packet delay profiles

identical (empirical) buffer overflow probabilities.

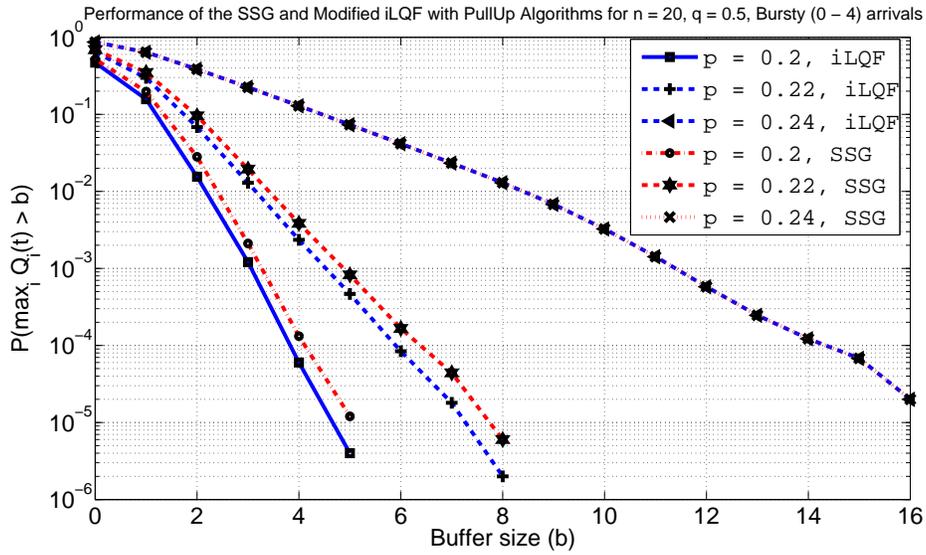


Figure 3.8: SSG v/s Modified iLQF with PullUp: Bursty load, buffer overflow probabilities

All the simulation results presented in this section vouch for the conclusion that the SSG and the Modified iLQF with PullUp algorithms perform very similar to each other, and consistently better than the MaxWeight algorithm, under a variety of arrival and channel processes.

## Chapter 4

### Multi-rate Channel Analysis

In this chapter, we again consider the problem of designing scheduling algorithms for multi-channel wireless downlink networks. As before, the main objective is to investigate the delay characteristics of these systems, under the assumption that it has a large number of users and a proportionally large bandwidth. The well-known MaxWeight-type algorithms [30] stabilize the system under a very general class of arrival and channel processes (such as Assumption 3.1), if there is any other scheduling algorithm than can do so. However, it was shown in Chapter 3, Section 3.2 that the MaxWeight algorithm results in a very poor delay performance for the system under consideration. We then proposed an algorithm called SSG (Server-Side Greedy) that, in addition to being throughput-optimal, results in a very good per-user delay performance. In Chapter 2, we have shown that a class of iterative scheduling algorithms, iLQF, gives the optimal small-delay performance in a large deviations sense under certain technical conditions. We also exhibited a particular algorithm in the iLQF class, namely iLQF with PullUp, that is optimal for the small delay problem under consideration.

The proofs of the good delay performance of the SSG algorithm and

the iLQF with PullUp algorithm crucially depended upon a sample-path dominance property of the algorithms: if there are two queuing systems with queues  $Q_i$  and  $R_i$ , with sample-path coupled arrivals and channels, and  $Q_i(t-1) \leq R_i(t-1)$  for all  $i$  and for some  $t$ , and both the systems use the same (SSG, or iLQF with PullUp) scheduling rule, then  $Q_i(t) \leq R_i(t)$  for all  $i$ . That is, if a queuing system dominates the other queuing system queue-by-queue, and if both the systems use the same (SSG, or iLQF with PullUp) scheduling rule, then the dominance continues to hold for all the future timeslots. This sample-path property fails to hold for the case when the channel service rates are more general than 0 or 1 packets per timeslot, rendering the earlier analysis techniques useless for the analysis of these more general systems. In this chapter, we present a new framework for analyzing the rate-function performance of the SSG and iLQF with PullUp-like algorithms that do not necessarily have the sample-path dominance property.

As before, we consider the same multi-queue multi-server discrete-time queuing system and require that the proposed algorithm(s) stabilize the system (make each of the queues positive recurrent) under Assumption 3.1. In addition, we require that our proposed algorithm(s) result in a strictly positive value of the rate function for the small-buffer overflow event (defined in Section 2.1) under the following two assumptions, if there is any other algorithm that can do so:

**Assumption 4.1** (Multi-level Channels and Arrivals).

*The number of packet arrivals to queue  $Q_i$  in timeslot  $t$  is the random variable*

$A_i(t)$ , where

$$A_i(t) = \begin{cases} 0 & \text{w.p. } p_0, \\ 1 & \text{w.p. } p_1, \\ \vdots & \\ M & \text{w.p. } p_M \end{cases}$$

where  $M \geq 1$  is an integer. In timeslot  $t$ , the server  $S_j$  can potentially serve  $X_{ij}(t)$  packets from  $Q_i$ , where  $X_{ij}(t)$  are modeled as random variables with

$$X_{ij}(t) = \begin{cases} 0 & \text{w.p. } q_0, \\ 1 & \text{w.p. } q_1, \\ \vdots & \\ K & \text{w.p. } q_K =: q \end{cases}$$

with  $q_i \in (0, 1)$  for all  $i$ , and  $\sum_i q_i = 1$ . We assume that all the random variables  $A_i(t)$  and  $X_{jk}(s)$  are mutually independent for all possible values of the involved parameters, that  $p_i > 0$  for all  $i \in \{0, 1, \dots, M\}$ ,  $\sum_{m=0}^M p_m = 1$ , and  $\sum_{m=1}^M mp_m < K$ .  $\diamond$

**Assumption 4.2** (Time-correlated Channel States).

The number of packet arrivals to queue  $Q_i$  in timeslot  $t$  is the random variable  $A_i(t)$ , where

$$A_i(t) = \begin{cases} M & \text{w.p. } p, \\ 0 & \text{w.p. } 1 - p, \end{cases}$$

where  $M \geq 1$  is an integer with  $pM < 1$ . In timeslot  $t$ , the server  $S_j$  can potentially serve  $X_{ij}(t)$  packets from  $Q_i$ , where  $X_{ij}(t)$  are modeled as Bernoulli random variables and are assumed to form a Markov chain for each pair  $(i, j)$ .

Specifically, if  $X_{ij}(t-1) = 0$ , then

$$X_{ij}(t) = \begin{cases} 1 & \text{w.p. } \alpha, \\ 0 & \text{w.p. } 1 - \alpha, \end{cases}$$

and if  $X_{ij}(t-1) = 1$ , then

$$X_{ij}(t) = \begin{cases} 1 & \text{w.p. } 1 - \beta, \\ 0 & \text{w.p. } \beta, \end{cases}$$

with  $\alpha, \beta \in (0, 1)$ . The arrival process is independent of the channel process. The channel-process Markov chains corresponding to different values of the pair  $(i, j)$  are mutually independent.  $\diamond$

Figure 4.1 shows the Markov chain corresponding to channel states for Assumption 4.2. Note that Assumptions 4.1 and 4.2 are restricted versions of Assumption 3.1.

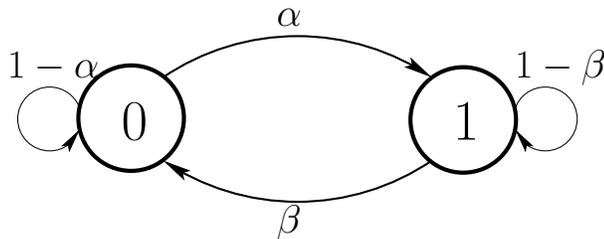


Figure 4.1: Markov Chain for Correlated Channels

To summarize, our objective is to design a scheduling algorithm that is throughput-optimal under Assumption 3.1, and results in a strictly positive value of the rate function under Assumption 4.1 and (separately) under Assumption 4.2, if there is any other algorithm that can do so.

## 4.1 Preliminary Analysis

In this section, we present certain basic results regarding the stability of the system under Assumptions 4.1 and 4.2, and also algorithm-independent

upper bounds on the rate-function under these assumptions.

**Lemma 4.1.** *Under Assumption 4.1, if  $\sum_{m=1}^M mp_m > K$ , then the system is unstable under any scheduling algorithm. If  $\sum_{m=1}^M mp_m < K$ , then there exists a constant  $n_0 = n_0(p, M, K, q)$  such that for all  $n \geq n_0$ , the system is stable under some algorithm.*

*Proof.* Please see Appendix C.1. □

**Theorem 4.1.** *Under Assumption 4.1, for any scheduling rule,*

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq p_M^{\lfloor \frac{b}{M} \rfloor + 1} (q_0)^{n(\lfloor \frac{b}{M} \rfloor + 1)}.$$

*Consequently, under any scheduling rule,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \leq \left( \left\lfloor \frac{b}{M} \right\rfloor + 1 \right) \log \frac{1}{q_0}.$$

*Proof.* Please see Appendix C.2. □

**Lemma 4.2.** *Under Assumption 4.2, if  $pM > 1$ , then the system is unstable under any scheduling algorithm. If  $pM < 1$ , then there exists a constant  $n_0 = n_0(p, M, \alpha, \beta)$  such that for all  $n \geq n_0$ , the system is stable under some algorithm.*

*Proof.* Please see Appendix C.3. □

**Theorem 4.2.** *Under Assumption 4.2, for any scheduling rule,*

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \geq p^{\lfloor \frac{b}{M} \rfloor + 1} (\min(\beta, 1 - \alpha))^n (1 - \alpha)^{n \lfloor \frac{b}{M} \rfloor}.$$

*Consequently, under any scheduling rule,*

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \leq \log \frac{1}{\min(\beta, 1 - \alpha)} + \left\lfloor \frac{b}{M} \right\rfloor \log \frac{1}{1 - \alpha}.$$

*Proof.* Please see Appendix C.4. □

**Lemma 4.3.** *Under Assumption 4.1, if*

$$\sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil > 1,$$

*then under any scheduling algorithm, we have*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > 0\right) = 0.$$

*Proof.* Please see Appendix C.5. □

**Remark 4.1.** *It is possible that under Assumption 4.1, for some constant  $b$  large enough, and under some algorithm, we have*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) > 0.$$

*However, since we are primarily concerned with the small-buffer overflow event, for the purpose of this dissertation, we are not interested in the case where the system parameters are such that the rate-function is zero at a finite value of  $b$  (even at  $b = 0$ ). We therefore assume throughout the rest of this dissertation that  $\sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil < 1$  whenever we mention Assumption 4.1.*

## 4.2 Stochastic Dominance of Markov Chains

The next three technical theorems are instrumental in characterizing the rate-function performance of the SSG and iLQF with PullUp-like algorithms and are interesting in their own right.

**Theorem 4.3.** *Consider a discrete-time, multi-dimensional Markov chain  $\mathbf{X}(t) = [X_1(t), X_2(t), \dots, X_n(t)]$ . Let  $\mathbf{X}(t)$  take values on the countable state-space  $\mathbb{Z}_+^n$ , and let the corresponding transition probabilities be*

$$p(\mathbf{x}, \mathbf{y}) := \mathbb{P}(\mathbf{X}(t+1) = \mathbf{y} \mid \mathbf{X}(t) = \mathbf{x}).$$

*Suppose that the Markov chain  $\mathbf{X}(t)$  has a stationary distribution  $\sigma(\mathbf{x})$ . Further, let*

$$\mathcal{W}_k := \left\{ [x_1, x_2, \dots, x_n] : \max_{1 \leq i \leq n} x_i = k \right\}.$$

*Then a stationary distribution of  $X^*(t) := \max_{1 \leq i \leq n} X_i(t)$  is given by a stationary distribution of the (one-dimensional) Markov chain  $Y(t)$ , taking values in  $\mathbb{Z}_+$ , whose transition probabilities are given by*

$$\mathbb{P}(Y(t+1) = y \mid Y(t) = x) = \mathbb{P}(\mathbf{X}(t+1) \in \mathcal{W}_y \mid \mathbf{X}(t) \in \mathcal{W}_x),$$

*where the conditional probability term on the RHS is under the stationary distribution  $\sigma(\cdot)$ .*

*Proof.* Please see Appendix C.6. □

The above theorem gives us a way to calculate the stationary distribution of the maximum value of a multi-dimensional Markov chain (which on its

own does not have the Markov property) by thinking of it as a one-dimensional Markov chain. Note that we do not need the stationary distribution to be unique for either  $\mathbf{X}(t)$  or  $Y(t)$ , and also that the result holds (with trivial modifications in the notation) for the Markov chains on  $\mathbb{Z}^n$ . Note also that if the Markov chain  $\mathbf{X}(t)$  is irreducible and has a stationary distribution, then the stationary distribution is unique (Theorem 3.1, Chapter 3 in [5]).

**Theorem 4.4.** *Consider two discrete-time Markov chains  $Y(t)$  and  $W(t)$  evolving on the same state-space  $\mathbb{Z}_+$ . Let the random variable  $Z_t(i)$  denote the increment in  $Y(t)$  when  $Y(t) = i$ . (Note that  $Z_t(i)$  can be negative.) Similarly, let  $\tilde{Z}_t(j)$  denote the increment in  $W(t)$  when  $W(t) = j$ . Let  $Z_t(i) \leq_{st} \tilde{Z}_t(j)$  for all  $i, j, t \in \mathbb{Z}_+$ , and  $Y(0) \leq_{st} W(0)$ . Then  $Y(t) \leq_{st} W(t)$  for all  $t \geq 0$ . In particular, if  $Y(t)$  and  $W(t)$  are ergodic (aperiodic, irreducible, positive recurrent) and have stationary distributions  $\mu_Y(\cdot)$  and  $\mu_W(\cdot)$  respectively, and the random variables  $Y, W$  are distributed according to  $Y \sim \mu_Y(\cdot)$  and  $W \sim \mu_W(\cdot)$ , then  $Y \leq_{st} W$ .*

*Proof.* Please see Appendix C.7. □

We now analyze the steady-state distribution of a special class of one-dimensional Markov chains from a rate-function point of view. The Markov chains in this class are similar to birth-death Markov chains, except that there can be multiple (but a finite, bounded number of) “births” in a given timeslot, or at most one “death.” The probability of birth(s) is “small,” and the probability of death is at least a constant. Hence, it is reasonable to expect that

the stationary distribution is strongly concentrated around 0. We quantify this intuition in a large-deviations sense in the following theorem.

**Theorem 4.5.** *Consider a family of Markov chains  $W^{(n)}(t)$  on the set  $\mathbb{Z}_+$ , having the following transition probability structure: there exists an integer  $n_0$  such that for all  $n \geq n_0$ , for all  $x \in \mathbb{Z}_+$ , for some fixed integer  $F$  and positive real numbers  $c, \eta$  we have*

$$\mathbb{P}(W(t+1) = x+k \mid W(t) = x) = \eta e^{-cnk} \text{ for } 1 \leq k \leq F,$$

and

$$\mathbb{P}(W(t+1) = x-1 \mid W(t) = x, x > 0) = \frac{1}{2}.$$

Then there exists an integer  $n_1$  such that for all  $n \geq n_1$ , the Markov chain  $W^{(n)}(t)$  is positive recurrent, and for any integer  $s \geq 0$ , we have

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}(W^{(n)}(0) > s) \geq c(s+1),$$

where  $\mathbb{P}(\cdot)$  denotes the stationary distribution of  $W^{(n)}(t)$ .

*Proof.* Please see Appendix C.8. □

Figure 4.2 shows an example of a Markov chain referred to in Theorem 4.5, with  $F = 2, \eta = 1$ , and where the “self-loops” for the transition probabilities are not shown for simplicity. Theorem 4.5 says that the stationary distribution of this Markov chain is given by  $\pi_m \approx \pi_0 e^{-cnm}$ , in a large deviations sense as  $n \rightarrow \infty$ . In other words, the Markov chain is very similar to the birth-death Markov chain in that the distribution is approximately geometric.

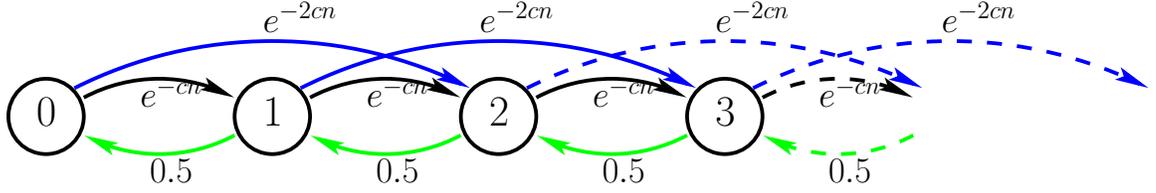


Figure 4.2: A candidate Markov chain for the rate-function calculation in Theorem 4.5

### 4.3 Analysis of the SSG Scheduling Rule

The SSG (Server-Side Greedy) scheduling algorithm was introduced in [4] (also see Definition 3.3, Chapter 3). This scheduling rule is interesting because of the following reasons:

1. It is throughput-optimal for the system under Assumption 3.1 (Theorem 3.5, Chapter 3).
2. It yields a strictly positive rate-function for the system under Assumption 4.1 with  $K = 1$  (Theorem 3.7, Chapter 3).
3. Its computational complexity ( $\mathcal{O}(n^2)$  computations per timeslot) is comparable to that of the MaxWeight rule ( $\Omega(n^2)$  computations per timeslot), but the MaxWeight rule yields a zero rate-function for all integers  $b \geq 0$  for the system under Assumption 4.1 with  $K = 1$  (Theorems 3.4 and 3.8, Chapter 3).

We now analyze the rate-function performance of the SSG rule under Assumption 4.1 by showing that:

1. In any given timeslot, and starting with any configuration of queue-lengths, the maximum effective queue-length increases (from its starting value, before the arrivals) with a very small probability.
2. In a constant  $k_0$  number of timeslots, the maximum effective queue-length decreases with at least a constant probability.

We then use Theorems 4.3, 4.4 and 4.5 to conclude the positivity of the rate-function. We first present a technical lemma that demonstrates a crucial property of the SSG rule. Note that this is a deterministic property of the SSG rule, and it does not require the number of queues or servers to be large in order to be true. This property establishes a stronger version of the following statement: if we have  $m$  queues, *each* connected to  $m$  servers with a channel of rate  $= K$ , then the maximum queue-length decreases by at least  $K$  (or becomes 0) at the end of service in that timeslot.

**Lemma 4.4.** *Under Assumption 4.1, let the set of queues be  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_m\}$ , the set of servers be  $\mathcal{S} = \{S_1, S_2, \dots, S_w\}$ . Fix a timeslot  $t$ , and let the queue-lengths after arrivals in that timeslot be  $L_1, L_2, \dots, L_m$ . Consider the bipartite graph  $G(\mathcal{Q} \cup \mathcal{S}, \mathcal{E})$  where  $\mathcal{E}$  denotes the set of edges, where an edge is present between a queue  $Q_i$  and a server  $S_j$  if the corresponding channel supports a rate  $= K$  in the timeslot  $t$ . Suppose further that every queue  $Q_i$  is connected to at least  $xm$  servers for some integer  $x \geq 1$ . Then with the system implementing the SSG rule, for every  $i \in \{1, 2, \dots, m\}$ , we have*

$$Q_i(t) \leq \max_{1 \leq j \leq m} (L_j - xK)^+.$$

*Proof.* Please see Appendix C.9.  $\square$

Let  $\zeta(t) := \max_{1 \leq i \leq n} Q_i(t)$  denote the maximum queue-length in the system at the end of timeslot  $t$ .

**Lemma 4.5.** *Let Assumption 4.1 hold with  $r = \left\lceil \frac{M}{K} \right\rceil$  for some integer  $r \geq 1$ .*

*Fix*

$$\epsilon \in \left( 0, \min \left( \frac{p_0}{M}, \frac{1 - \sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil}{rM} \right) \right),$$

*and let*

$$\mathcal{B}_\epsilon := \{[x_1, x_2, \dots, x_{M+1}] \subseteq \mathfrak{R}^{M+1} : |x_i| < \epsilon \quad \forall i\}$$

*and*

$$\tau := \inf_{z \in \mathcal{N}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}} \sum_{m=0}^M z_m \log \frac{z_m}{p_m}.$$

*Then for a system under the SSG rule, for any fixed  $\rho > 0$ , for  $n$  large enough, and for any timeslot  $t$ , we have*

$$\mathbb{P}(\zeta(t) > \zeta(t-1)) \leq e^{-n\tau(1-\rho)} + Mn(1-q)^{n\delta} + Mn\delta \exp\left(-\frac{2n\delta}{q} H\left(\frac{q}{2} \mid q\right)\right).$$

*Proof.* Please see Appendix C.10.  $\square$

**Lemma 4.6.** *Let Assumption 4.1 hold. Fix any timeslot  $t$ , and let  $\zeta(t) = k$  for some integer  $k$ . Then, for a system using the SSG rule, there exists a constant*

integer  $k_0$  such that for all  $n$  large enough, we have

$$\mathbb{P}(\zeta(t + k_0) < k \mid \zeta(t) = k, k > 0) \geq \frac{1}{2}.$$

*Proof.* This proof is on the same lines as that of Lemma 3.9 and has been omitted to avoid repetition.  $\square$

Now we are in a position to quantify the rate-function performance of the SSG algorithm.

**Theorem 4.6.** *Let Assumption 4.1 hold with  $r = \left\lceil \frac{M}{K} \right\rceil$  for some integer  $r \geq 1$ . Fix*

$$\epsilon \in \left( 0, \min \left( \frac{p_0}{M}, \frac{1 - \sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil}{rM} \right) \right),$$

and let

$$\mathcal{B}_\epsilon := \{[x_1, x_2, \dots, x_{M+1}] \subseteq \mathfrak{R}^{M+1} : |x_i| < \epsilon \quad \forall i\}$$

and

$$\tau := \inf_{z \in \mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}} \sum_{m=0}^M z_m \log \frac{z_m}{p_m}.$$

Then for a system using the SSG rule, for any integer  $b \geq 0$ , we have

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \frac{b+1}{M} \min \left( \tau, \delta \log \frac{1}{1-q}, \frac{2\delta H\left(\frac{q}{2} \mid q\right)}{q} \right) > 0.$$

*Proof.* Please see Appendix C.11.  $\square$

We now study the performance of the SSG algorithm under Assumption 4.2. Under this assumption, regardless of the channel realizations in the previous timeslot(s), any given channel is ON in the current timeslot  $t$  with probability at least  $\gamma = \min(\alpha, 1 - \beta)$ . In view of this observation, the proof of the following theorem is on the same lines as that of Theorem 4.6.

**Theorem 4.7.** *Let Assumption 4.2 hold. Define  $\gamma := \min(\alpha, 1 - \beta)$ . Fix any constant  $p' \in (p, 1)$  and  $\delta \in \left(0, \min\left(p', \frac{\gamma(1-p')}{2-\gamma}\right)\right)$ . In particular, let  $p' = (1+p)/2$  and  $\delta = \min\left(\frac{p'}{2}, \frac{\gamma(1-p')}{2}\right)$ . Then for a system using the SSG rule, for any integer  $b \geq 0$ , we have*

$$\begin{aligned} & \liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \\ & \geq \frac{b+1}{M} \min \left( H(p'|p), \delta \log \frac{1}{1-\gamma}, \frac{2\delta H\left(\frac{\gamma}{2}|\gamma\right)}{\gamma} \right) > 0. \end{aligned}$$

*Proof.* The proof of this theorem is on the same lines as those of Theorem 4.6. We omit the details. □

An immediate strengthening of these rate-function results can be obtained by optimizing the RHS over the appropriate ranges for  $\epsilon, p'$  and  $\delta$ .

## 4.4 The DMEQ Scheduling Rule

So far, we have discussed algorithms (SSG, MaxWeight) that have limited computational complexity (polynomial in the description length of the

system) and analyzed their throughput and rate-function performance. At the other extreme, we ignore the computational complexity considerations and ask the question: what is the best rate-function performance an algorithm can give? The answer to this question leads us to the characterization of sufficient conditions for rate-function optimality under certain conditions that the author believes are also “necessary” in some sense. In this section, we study the performance of an algorithm, DMEQ, which stands for DeepestDrain of Maximum Effective Queue-length. This algorithm provides the optimal rate-function performance for the small-buffer overflow event under certain conditions, and it uses the DeepestDrain operation (defined next) as a “black-box.”

The definition of the DeepestDrain operation refers to the term “the effective queue-length.” As the name suggests, the effective length of a queue is a constant multiple of its actual length, where different queues can have different multipliers for the effective queue-length calculation. This mathematical model is useful for the case when the different queues have incoming flows with different delay requirements. For the purpose of this exposition, we consider all the queues having the same multiplier = 1, i.e., *the queue-length is the same as the effective queue-length*. We nevertheless retain the name “effective queue-length” to remind the reader that significant generalizations of the model are possible.

**Definition 4.1** (The DeepestDrain Operation). *The DeepestDrain operation computes an allocation of the servers to the queues as follows:*

**Input:**

1. The queue-length vector  $\underline{L} = [L_1, L_2, \dots, L_m]$  for the queues  $\underline{Q} = [Q_1, Q_2, \dots, Q_m]$ .
2. The server vector  $\underline{S} = [S_1, S_2, \dots, S_k]$ .
3. The channel-rate matrix  $\mathbf{X} = [X_{ij}]$  for  $1 \leq i \leq m, 1 \leq j \leq k$ .

**Steps:** Find an allocation  $\mathbf{Y} = [Y_{ij}]$  of servers to the queues that obeys the following conditions, breaking ties between multiple candidate allocations arbitrarily:

1. Minimize the maximum effective queue-length (MEQ).
2. Among all the allocations that result in the minimum value of the MEQ (after allocation), choose the one that minimizes the number of queues at the MEQ.
3. Among all the allocations that satisfy the conditions so far, choose the one that minimizes the second MEQ.
4. Among all the allocations that satisfy the conditions so far, choose the one that minimizes the number of queues with the second MEQ.
- $\vdots$
- $2m - 1$ . Among all the allocations that satisfy the conditions so far, choose the one that minimizes the  $m^{\text{th}}$  MEQ.

2m. Among all the allocations that satisfy the conditions so far, choose the one that minimizes the number of queues with the  $m^{\text{th}}$  MEQ.

**Output:** The allocation matrix  $\mathbf{Y}$ , where  $Y_{ij} = 1$  if server  $S_j$  is allocated to queue  $Q_i$ , and 0 otherwise.  $\diamond$

Note that the DeepestDrain operation finds an allocation that *simultaneously* satisfies all the conditions mentioned in the definition, possibly through a brute-force search over all possible schedules. We do not know of an explicit algorithm that solves this optimization problem in polynomial time. Nevertheless, the DeepestDrain scheme produces a valid schedule.

We write  $\mathbf{Y} = \text{DeepestDrain}(\underline{Q}, \underline{L}, \underline{S}, \mathbf{X})$  for the above operation. The number of steps (conditions) that the operation needs to check can be less than  $2m$ , depending upon if all the servers have been allocated, etc. In such a case, the remaining conditions are trivially satisfied by the candidate allocation. We now define a server allocation policy (DMEQ) that uses the DeepestDrain operation.

**Definition 4.2** (DeepestDrain of Maximum Effective Queue-length : DMEQ). *The DMEQ rule allocates servers to queues in timeslot  $t$  according to the following procedure.*

**Input:**

1. The queue-lengths  $Q_i(t-1)$  for  $1 \leq i \leq n$ .
2. The arrival vector  $A_i(t)$  for  $1 \leq i \leq n$ .

3. The channel rates  $X_{ij}(t)$  for  $1 \leq i, j \leq n$ .

**Steps:**

1. Compute the queue-lengths after arrivals,  $L_i(t) := Q_i^{(0)}(t) = Q_i(t-1) + A_i(t)$  for all  $i$ .

2. For all  $j \in \{1, 2, \dots, n\}$ , define

$$\mathcal{J}_j := \arg \max_{1 \leq i \leq n} Q_i^{(0)}(t) X_{ij}(t),$$

$$\text{and } c_j := \min \left[ \arg \max_{m \in \mathcal{J}_j} X_{mj}(t) \right].$$

For any server  $S_j$  and all  $i$ , if  $X_{ij}(t) < X_{c_j j}(t)$ , then redefine  $X_{ij}(t) = 0$ , and use this updated value of  $X_{ij}(t)$  throughout the rest of the timeslot  $t$ .

3. Define the following quantities:

(a)  $\underline{Q} = [Q_1, Q_2, \dots, Q_n]$ .

(b)  $\underline{L} = [L_1(t), L_2(t), \dots, L_n(t)]$ .

(c)  $\underline{S} = [S_1, S_2, \dots, S_n]$ .

(d)  $\mathbf{X} = [X_{ij}(t)]$  for  $1 \leq i, j \leq n$ , using the updated values as per the Step 2.

4. Find an allocation  $\mathbf{B} = \text{DeepestDrain}(\underline{Q}, \underline{L}, \underline{S}, \mathbf{X})$ .

5. Define  $Y_{ij}(t) = B_{ij}$ . Compute the resultant queue-lengths  $Q_i(t) = (L_i(t) - \sum_{j=1}^n X_{ij}(t) Y_{ij}(t))^+$ .

**Output:**

1. The server allocations,  $Y_{ij}(t)$  for  $1 \leq i, j \leq n$ .
2. The queue-lengths  $Q_i(t)$  for  $1 \leq i \leq n$ . ◇

Note that DMEQ is a class of rules as a result of the arbitrary tie-breaking in the definition of the DeepestDrain operation.

#### 4.4.1 Throughput-optimality of DMEQ

Our first aim is to establish the throughput-optimality of the DMEQ rule. The following lemma shows that in every timeslot, the weight of the schedule under the MaxWeight rule is at most an additive constant more than that under the DMEQ rule.

**Lemma 4.7.** *Fix any DMEQ-class rule  $\Lambda$ . Consider any timeslot  $t$ , and let the queue-lengths immediately after arrivals in that timeslot be  $\ell_i$ , i.e.,  $\ell_i = Q_i(t-1) + A_i(t)$ . Let  $c_j$  be as defined in step 2 in the definition of the DMEQ-class rules (Definition 4.2). Then,  $W^{DMEQ}(t) \geq W^{MW}(t) - n^2 \hat{\mu}^2$ .*

*Proof.* Please see Appendix C.12. □

As in the case of Theorem 3.1, the throughput-optimality of the DMEQ-class rules is now immediate.

**Theorem 4.8** (Throughput-optimality of DMEQ). *Under Assumption 3.1 on the arrival and channel processes, any DMEQ-class rule makes the system*

stable in the mean, i.e.,

$$\limsup_{p \rightarrow \infty} \frac{1}{p} \sum_{k=0}^{p-1} \mathbb{E} \left[ \sqrt{\sum_{i=1}^n Q_i^2(k)} \right] < \infty.$$

In addition, if the arrival and channel state processes are such that the DMEQ rule makes the queuing system an aperiodic Markov chain with a single communicating class, then the stability in the mean implies that the Markov chain is positive recurrent [20].

*Proof.* In view of Lemma 4.7, this proof is identical to that of Theorem 3.5 and has been omitted.  $\square$

#### 4.4.2 Rate-function Performance of DMEQ under Assumption 4.1

We now analyze the rate-function performance of the proposed DMEQ scheduling rule under Assumption 4.1. The next definition is concerned with the existence of a certain structure called an  $r$ -fold perfect matching in a large bipartite graph, and is instrumental in understanding the rate-function performance of the DMEQ rule.

**Definition 4.3** ( $r$ -fold perfect matching). *Consider a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ , where  $\mathcal{U} \cup \mathcal{V}$  denotes the set of nodes and  $\mathcal{E}$  denotes the set of edges. Let  $|\mathcal{V}| \leq r|\mathcal{U}|$  for some integer  $r \geq 1$ . Every edge  $e \in \mathcal{E}$  has one endpoint each in the sets  $\mathcal{U}$  and  $\mathcal{V}$ . A subset  $\mathcal{M} \subseteq \mathcal{E}$  is said to be an  $r$ -fold perfect matching in  $G$  with respect to  $\mathcal{U}$  if:*

1. For every node  $u \in \mathcal{U}$ , out of all the edges that have  $u$  as an endpoint, exactly  $r$  edges belong to  $\mathcal{M}$ .
2. For every node  $v \in \mathcal{V}$ , out of all the edges that have  $v$  as an endpoint, at most 1 edge belongs to  $\mathcal{M}$ .  $\diamond$

We refer to a 1-fold perfect matching as a perfect matching (with respect to  $\mathcal{U}$ ). Note that when  $|\mathcal{U}| = |\mathcal{V}|$ , a 1-fold perfect matching w.r.t.  $\mathcal{U}$  is a perfect matching in the graph. An example of a 2-fold perfect matching with respect to  $\mathcal{U}$  is shown in Figure 4.3, where the solid edges belong to the 2-fold perfect matching  $\mathcal{M}$  and the dotted edges belong to  $\mathcal{E} \setminus \mathcal{M}$ .

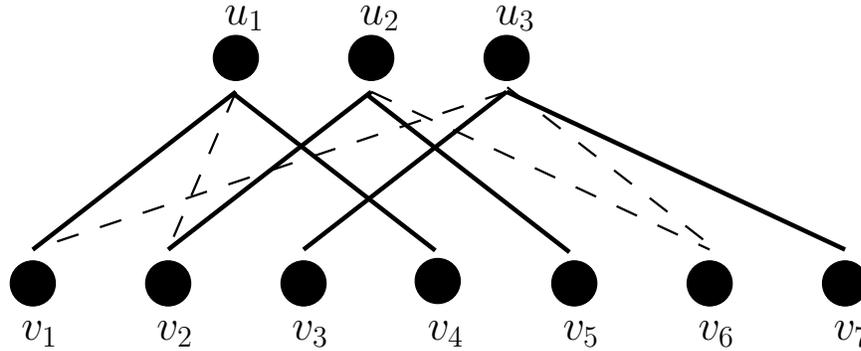


Figure 4.3: Example of a 2-fold perfect matching w.r.t.  $\mathcal{U}$

The following result regarding the existence of  $r$ -fold perfect matchings in large random bipartite graphs is a simple extension of Lemma 2.1.

**Lemma 4.8.** *Consider a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$  where  $|\mathcal{U}| = n$  and  $|\mathcal{V}| = rn$ , for some fixed integer  $r \geq 1$ . For  $u \in \mathcal{U}$  and  $v \in \mathcal{V}$ , define the indicator*

random variables

$$\chi_{uv} = \begin{cases} 1 & \text{if } (u, v) \in \mathcal{E}, \\ 0 & \text{if } (u, v) \notin \mathcal{E}. \end{cases}$$

Let  $\chi_{uv}$  be i.i.d. random variables with  $\mathbb{P}(\chi_{uv} = 1) = q$ . Then, for  $n$  large enough,

$$(1 - q)^n \leq \mathbb{P}(G \text{ has no } r\text{-fold perfect matching w.r.t. } \mathcal{U}) \leq 3rn(1 - q)^n.$$

*Proof.* Please see Appendix C.13. □

Let  $\zeta(t) = \max_{1 \leq i \leq n} Q_i(t)$  denote the maximum (effective) queue-length at the end of timeslot  $t$ .

**Lemma 4.9.** *Let Assumption 4.1 hold with  $r := \left\lceil \frac{M}{K} \right\rceil$ . Consider a queuing system that employs a DMEQ-class rule  $\Lambda$  for server allocation. Fix any timeslot  $T$ . After arrivals in timeslot  $T$ , let  $\mathcal{C}_m$  denote the set of queues with exactly  $m$  arrivals. For  $1 \leq w \leq r$ , let*

$$\mathcal{D}_w := \mathcal{C}_{(w-1)K+1} \cup \mathcal{C}_{(w-1)K+2} \cup \cdots \cup \mathcal{C}_{wK},$$

and let  $\mathcal{S} = \bigcup_{w=1}^r \mathcal{S}_w$  be a partition of the set of servers. Let  $\mathcal{E}_w$  denote the restriction of the set of edges to the set of nodes  $\mathcal{D}_w \cup \mathcal{S}_w$ , i.e., for  $Q_i \in \mathcal{D}_w, S_j \in \mathcal{S}_w$ , if the channel rate  $X_{ij} = K$ , then the corresponding edge is present in the set  $\mathcal{E}_w$ . Suppose further that the bipartite graph  $G_w(\mathcal{D}_w \cup \mathcal{S}_w, \mathcal{E}_w)$  has a  $w$ -fold perfect matching w.r.t.  $\mathcal{D}_w$ . Then  $\zeta(T) \leq \zeta(T - 1)$ .

*Proof.* Please see Appendix C.14. □

We are now ready to analyze the rate-function performance of the DMEQ algorithm. Our first aim is to show that under the DMEQ rule and for  $n$  large enough, the MEQ of the system increases in a given timeslot with very small probability.

**Lemma 4.10.** *Let Assumption 4.1 hold with  $r := \left\lceil \frac{M}{K} \right\rceil$ . Fix*

$$\epsilon \in \left( 0, \min \left( \frac{p_0}{M}, \frac{1 - \sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil}{rM} \right) \right),$$

and let

$$\mathcal{B}_\epsilon := \{[x_1, x_2, \dots, x_{M+1}] \subseteq \mathfrak{R}^{M+1} : |x_i| < \epsilon \quad \forall i\}$$

and

$$\tau := \inf_{z \in \mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}} \sum_{m=0}^M z_m \log \frac{z_m}{p_m}.$$

Let

$$[p'_0, p'_1, \dots, p'_M] = [p_0 - M\epsilon, p_1 + \epsilon, p_2 + \epsilon, \dots, p_M + \epsilon].$$

Then for a system using the DMEQ rule, for any  $\rho > 0$ , for  $n$  large enough, and for any timeslot  $t$ , we have

$$\mathbb{P}(\zeta(t) > \zeta(t-1)) \leq 3n \sum_{m=1}^M (1-q)^{p'_m} + e^{-n\tau(1-\rho)}.$$

*Proof.* Please see Appendix C.15. □

**Lemma 4.11.** *Let Assumption 4.1 hold. Then for a system using the DMEQ rule, there exists a constant integer  $k_0$  such that for all  $n$  large enough, and for any timeslot  $t$ , we have*

$$\mathbb{P}(\zeta(t + k_0) < \zeta(t) \mid \zeta(t) > 0) \geq \frac{1}{2}.$$

*Proof.* For ease of notation, we prove this claim only for the case  $p_M = p > 0$ , and  $p_0 = 1 - p$ . The proof for the more general case is almost the same. We further assume that  $M = rK$  and  $q_0 = 1 - q$ . None of these assumptions is binding and we can easily extend the proof to the more general case through the introduction of extensive notation.

Here is an informal road-map of the proof. Let  $\psi(t)$  denote the number of queues with the maximum effective queue-length at the end of timeslot  $t$ . We first show that there exists a server allocation rule under which, if we have  $\zeta(t + \ell) = \zeta(t + \ell - 1)$  for a given  $\ell \in \{1, 2, \dots, k_0\}$ , then  $\psi(t + \ell) \leq (\psi(t + \ell - 1) - n\epsilon)^+$  with high probability, for some fixed  $\epsilon > 0$ . We also show that under this rule, the probability that  $\zeta(t + \ell) > \zeta(t + \ell - 1)$  is very small. Together, these two claims imply that under the candidate rule, the MEQ decreases with high probability in at most  $\left\lceil \frac{1}{\epsilon} \right\rceil$  timeslots. Next, we use this scheduling rule to argue about the performance of the DMEQ rule, since by the definition of the DMEQ rule, if there is any other rule that can give so much reduction in the MEQ, or the number of queues with the maximum

effective queue-length, then so must the DMEQ rule. For a detailed proof, please see Appendix C.16.  $\square$

As a result of Lemmas 4.10 and 4.11, the maximum (effective) queue-length (MEQ) in the system has the following behavior (for  $n$  large):

1. In a given timeslot, it increases with an extremely small probability. Further, in a given timeslot, it can increase by at most the number of arrivals to a given queue, which is a finite number, uniformly bounded in  $n$ .
2. Over a constant number of timeslots, it decreases (by 1 or more) with at least a constant probability.

It is therefore reasonable to expect that the MEQ in the system is very strongly concentrated around 0. Our goal now is to formalize this observation and make a statement about the probability distribution of the MEQ.

**Theorem 4.9.** *Let Assumption 4.1 hold with  $r := \left\lceil \frac{M}{K} \right\rceil$ . Fix*

$$\epsilon \in \left( 0, \min \left( \frac{p_0}{M}, \frac{1 - \sum_{m=1}^M p_m \left\lceil \frac{m}{K} \right\rceil}{rM} \right) \right),$$

and let

$$\mathcal{B}_\epsilon := \{[x_1, x_2, \dots, x_{M+1}] \subseteq \mathfrak{R}^{M+1} : |x_i| < \epsilon \quad \forall i\}$$

and

$$\tau := \inf_{z \in \mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}} \sum_{m=0}^M z_m \log \frac{z_m}{p_m}.$$

Let

$$[p'_0, p'_1, \dots, p'_M] = [p_0 - M\epsilon, p_1 + \epsilon, p_2 + \epsilon, \dots, p_M + \epsilon].$$

Then for a system using the DMEQ rule, for any integer  $b \geq 0$ ,

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \frac{b+1}{M} \min \left( \tau, \left( \min_{1 \leq i \leq M} p'_m \right) \log \frac{1}{1-q} \right) > 0.$$

*Proof.* In view of Lemmas 4.10 and 4.11, this proof is very similar to that of Theorem 4.6 and has been omitted.  $\square$

#### 4.4.3 Rate-function Performance of DMEQ under Assumption 4.1, with $M \leq K$

We consider the special case where in Assumption 4.1, the maximum number of arrivals ( $M$ ) to a queue in any given timeslot is less than or equal to the channel rate in the state  $K$ , and the channel-rates are either 0 or  $K$  packets per timeslot, i.e.,  $q_0 = 1 - q_K = 1 - q$ . In this case, it is possible to tighten the rate-function analysis of DMEQ and provide better lower bounds.

**Lemma 4.12.** *Let Assumption 4.1 hold with  $M \leq K$ . Then for a system using the DMEQ rule, for  $n$  large enough, and for any timeslot  $t$ , we have*

$$\mathbb{P}(\zeta(t) > \zeta(t-1)) \leq 3n(1-q)^n.$$

*Proof.* Please see Appendix C.17.  $\square$

**Lemma 4.13.** *Let Assumption 4.1 hold with  $M \leq K$ . Fix any timeslot  $t$ . Then for a system using the DMEQ rule, there exists a constant integer  $k_0$  such that for all  $n$  large enough, we have*

$$\mathbb{P}(\zeta(t + k_0) < \zeta(t) \mid \zeta(t) > 0) \geq \frac{1}{2}.$$

*Proof.* The proof is very similar to that of Lemma 4.11 and has been omitted to avoid repetition.  $\square$

**Theorem 4.10.** *Let Assumption 4.1 hold with  $M \leq K$ . Then for a system using the DMEQ rule, for any integer  $b \geq 0$ , we have*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \frac{b+1}{M} \log \frac{1}{1-q} > 0.$$

*Proof.* In view of Lemmas 4.12 and 4.13, the proof of this theorem is identical to that of Theorem 4.9 and has been omitted to avoid repetition.  $\square$

We know from Theorem 4.1 that  $\left( \left\lfloor \frac{b}{M} \right\rfloor + 1 \right) \log \frac{1}{1-q}$  is an upper bound on the rate-function under Assumption 4.1. Consider the case where  $b+1$  is divisible by  $M$ .<sup>1</sup> In this case,  $\left\lfloor \frac{b}{M} \right\rfloor + 1 = \frac{b+1}{M}$  and the proposed DMEQ algorithm is rate-function optimal for the values  $b = M-1, 2M-1, \dots$ . The author believes that the possible sub-optimality of the DMEQ algorithm

---

<sup>1</sup>For the case  $M = 1$ , this condition holds for all integers  $b \geq 0$ .

for the other values of  $b$  is a result of a slack in the proof, and that the DMEQ algorithm (or minor modifications of it) are rate-function optimal for all integers  $b \geq 0$ .

#### 4.4.4 Rate-function Performance of DMEQ under Assumption 4.2

We establish a lower bound on the rate-function performance of the proposed DMEQ algorithm under Assumption 4.2.

**Theorem 4.11.** *Let Assumption 4.2 hold. Then for a system using the DMEQ rule, for any integer  $b \geq 0$ , we have*

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \geq \frac{b+1}{M} \log \frac{1}{1 - \min(\alpha, 1 - \beta)} > 0.$$

*Proof.* Regardless of the channel realizations in the previous timeslot(s), any given channel is ON in the current timeslot  $t$  with probability at least  $\min(\alpha, 1 - \beta)$ . In view of this observation, the proof of this theorem is identical to that of Theorem 4.10 and has been omitted to avoid repetition.  $\square$

Since the channel-rates under Assumption 4.2 belong to the set  $\{0, 1\}$ , the iHLQF algorithm can be analyzed for this case, thanks to its sample-path-dominance property (Theorem 2.8), and the positivity of the rate-function can be proved. However, the DMEQ algorithm continues to give a provably positive rate-function for the small-buffer overflow event even if the time-correlated channels support rates other than just 0 or 1 packets per timeslot, and a number of immediate generalizations of the above result are possible.

## 4.5 Unequal Number of Queues and Servers

In this section, we consider the case where the system has  $n$  queues and  $\xi n$  servers. Here  $\xi \in [1, \infty)$  is a fixed constant, independent of  $n$ .<sup>2</sup> This range of values of  $\xi$  is of interest because in typical OFDM-based wireless downlink systems, the number of orthogonal channels is much larger than the number of active users.

**Theorem 4.12.** *Fix any constant  $\xi \in [1, \infty)$ , and a system with  $n$  queues and  $\xi n$  servers. Then the DMEQ and the SSG rule are throughput-optimal for the system under Assumption 3.1. Further, under Assumption 4.1 or 4.2, if  $\Gamma$  is a lower bound on the rate function for a symmetric (i.e.,  $n$  servers and  $n$  queues) system implementing the DMEQ (resp. SSG) rule, then  $\xi\Gamma$  is a lower bound on the rate function for the system with  $n$  queues and  $\xi n$  servers, implementing the DMEQ (resp. SSG) rule.*

*Proof.* Please see Appendix C.18. □

Similar rate-function positivity (and optimality) results can be derived for the iLQF with PullUp algorithm in Chapter 2 for systems with channel-rates confined to 0 or 1 packets per timeslot. This extension of results is possible because the sample-path-dominance property of the iLQF with PullUp

---

<sup>2</sup>We ignore the issues involving  $\xi n$  not being an integer. The reason is that in the limit as  $n \rightarrow \infty$ ,  $\xi n$  can be replaced by  $\lfloor \xi n \rfloor$  with cosmetic changes to the proofs. For finite  $n$ , the reader may consider a sequence  $\xi(n) = \lfloor \xi n \rfloor / n$  that approaches  $\xi$  such that  $n\xi(n)$  is an integer for all  $n$ , and a system with  $n\xi(n) = \lfloor \xi n \rfloor$  servers. Hereafter in this exposition, we do not discuss this issue.

algorithm does not depend upon the channel process statistics or the number of queues and servers, but only the fact that the channel-rates belong to the set  $\{0, 1\}$ . Although we do not consider the case  $\xi < 1$  here, the analysis techniques developed so far (in particular, Theorems 4.3, 4.4 and 4.5) can be used to analyze this case as well, implying rate-function positivity results for SSG and DMEQ under the appropriate stability conditions.

## Chapter 5

### Conclusions

We considered the problem of designing scheduling algorithms for multi-user multi-channel wireless downlink networks, with emphasis on good per-user delay performance in addition to network stability. We showed that the classic MaxWeight-type algorithms result in a very poor delay performance. We presented a class of algorithms called iLQF (iterated Longest Queues First) that provides the optimal delay performance under certain network conditions, and consistently good delay performance in a variety of network settings. The computational complexity of the iLQF-class algorithms can be prohibitive for real-time implementations. To overcome this issue, we presented a class of algorithms called SSG (Server-Side Greedy) that provides a trade-off between computational complexity and delay performance.

The proofs of the good delay performance of the iLQF and SSG algorithms crucially depend on a certain sample-path dominance property that fails to hold for all but the simplest of systems. To overcome this dependence, we presented properties of Markov chains that might be of independent interest, and used those to prove that our algorithms continue to give a good delay performance for a wide variety of systems. We finally provided sufficient

conditions for an algorithm to be delay-optimal for the more general systems, namely for the systems with multi-rate channels, under certain technical conditions. We validated our results through analysis and simulations.

The main intuition that emerges from this work is that in order to provide a good delay performance, one should allocate the resources in an iterative manner, taking into account the effect of prior allocations when making decisions for the current resource allocation. The MaxWeight-type algorithms argue that for network stability, in every timeslot, one should choose an allocation that maximizes the sum of the products of the queue-lengths and the serving channel-rates. Our results show that this intuition is valid only in a limited sense: for providing good delay performance, one should only approximately maximize the aforementioned sum, while paying attention to the finer queue-length dynamics and resource wastage. This kind of a “careful” resource allocation dramatically improves the per-user delay performance, in addition to providing throughput-optimality.

## Appendices

# Appendix A

## Proofs for Chapter 2

### A.1 Proof of Theorem 2.2

Consider a service rule where in each timeslot, each server uniformly and randomly picks a queue to which it has an ON channel, and serves it. If that particular chosen queue is empty, then that server does not serve any queue in that timeslot. (Multiple servers can serve the same queue, but there is no co-ordination between the servers.)

Then, the probability that the first server offers its service to the first queue in a particular timeslot is

$$\mathbb{P}(Y_{11}(t) = 1) = \mathbb{P}(Y_{11}(t) = 1 \mid X_{11}(t) = 1) \cdot \mathbb{P}(X_{11}(t) = 1).$$

Now, for the service rule under consideration,

$$\begin{aligned} \mathbb{P}(Y_{11}(t) = 1 \mid X_{11}(t) = 1) &= \sum_{j=0}^{n-1} \mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot } t \mid X_{11}(t) = 1, \\ &\quad \text{Exactly } j \text{ of the rest } n - 1 \text{ channels from } S_1 \text{ are ON}) \\ &\quad \cdot \mathbb{P}(\text{Exactly } j \text{ of the rest } n - 1 \text{ channels from } S_1 \text{ are ON}) \\ &= \sum_{j=0}^{n-1} \frac{1}{j+1} \binom{n-1}{j} q^j (1-q)^{n-1-j} \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=0}^{n-1} \frac{1}{j+1} \cdot \frac{(n-1)!}{j!(n-1-j)!} q^j (1-q)^{n-1-j} \\
&= \frac{1}{n} \sum_{j=0}^{n-1} \frac{n!}{(j+1)!(n-1-j)!} q^j (1-q)^{n-1-j} \\
&= \frac{1}{qn} \sum_{j=0}^{n-1} \binom{n}{j+1} q^{j+1} (1-q)^{n-1-j} \\
&= \frac{1 - (1-q)^n}{qn}.
\end{aligned}$$

Thus,  $\mathbb{P}(Y_{11}(t) = 1) = \frac{1 - (1-q)^n}{n}$ , implying that the total amount of service offered to the first queue (or to any other queue, by symmetry) in timeslot  $t$  is  $1 - (1-q)^n$ . If  $p < 1$  and  $q > 0$  are fixed, then  $1 - (1-q)^n > p$  for large enough  $n \geq n_0(p, q)$ , where

$$n_0(p, q) := \left\lceil \frac{\log(1-p)}{\log(1-q)} \right\rceil,$$

implying that all the queues are stable (positive recurrent) under the specified policy, for  $n$  large enough.

## A.2 Proof of Lemma 2.1

For  $\mathcal{A} \subseteq \mathcal{U}$ , let  $\Gamma(\mathcal{A})$  denote the neighborhood  $\mathcal{A}$ , i.e.,

$$\Gamma(\mathcal{A}) := \{b \in \mathcal{V} : (a, b) \in \mathcal{E} \text{ for some } a \in \mathcal{A}\}.$$

We know from Hall's theorem ([19], Thm. 7.40) that if a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$  does not have a perfect matching, then there exists a subset

$\mathcal{A} \subseteq \mathcal{U}$  such that  $|\Gamma(\mathcal{A})| < |\mathcal{A}|$ . Fix a nonempty subset  $\mathcal{A} \subseteq \mathcal{U}$  and a subset  $\mathcal{B} \subseteq \mathcal{V}$ . Let  $|\mathcal{A}| = a$ . Then we have

$$\begin{aligned} \mathbb{P}(\Gamma(\mathcal{A}) \subseteq \mathcal{B}) &= \mathbb{P}(\text{No node in } \mathcal{A} \text{ connects to any node in } \mathcal{S} \setminus \mathcal{B}) \\ &= (1 - q)^{(n - |\mathcal{B}|)a}. \end{aligned}$$

If the graph has no perfect matching, then by Hall's theorem, there must exist sets  $\mathcal{A}$  and  $\mathcal{B}$  such that

1.  $\mathcal{A} \subseteq \mathcal{U}$ ,  $\mathcal{B} \subseteq \mathcal{V}$ ,
2.  $|\mathcal{B}| = |\mathcal{A}| - 1$ ,
3.  $\Gamma(\mathcal{A}) \subseteq \mathcal{B}$ .

Hence, by union bound over all possible subsets  $\mathcal{A} \subseteq \mathcal{U}$  and all possible corresponding subsets  $\mathcal{B} \subseteq \mathcal{V}$ , we have

$$\begin{aligned} \mathbb{P}(G \text{ has no perfect matching}) &\leq \sum_{a=1}^n \binom{n}{a} \cdot \binom{n}{a-1} \cdot (1 - q)^{a(n-a+1)} \\ &\leq 2 \sum_{a=1}^{\lceil n/2 \rceil} \binom{n}{a} \cdot \binom{n}{a-1} \cdot (1 - q)^{a(n-a+1)}, \quad (\text{A.2.1}) \end{aligned}$$

where the last inequality holds with equality if  $n$  is even.

We consider the case when  $n$  is large, in particular  $n > 2$ . Now, for

$n > 2$  and  $1 < a \leq \lceil n/2 \rceil$ ,  $a - 1 \geq a/2$ ,  $n - a \geq n/3$ , and we have

$$\begin{aligned}
\frac{\binom{n}{a} \binom{n}{a-1} (1-q)^{a(n-a+1)}}{n(1-q)^n} &\leq \frac{n^a \cdot n^{a-1} \cdot (1-q)^{a(n-a+1)}}{n(1-q)^n} \\
&\leq n^{2a} (1-q)^{(n-a)(a-1)} \\
&\leq n^{2a} (1-q)^{na/6} \\
&= \exp\left(2a \log n - \frac{na}{6} \log \frac{1}{1-q}\right) \\
&= \exp\left[-\frac{a}{6} \left\{n \log \frac{1}{1-q} - 12 \log n\right\}\right] \\
&\leq \exp\left\{-\frac{a}{6} \cdot \frac{n}{2} \cdot \log \frac{1}{1-q}\right\}, \quad \text{for } n \text{ large enough} \\
&\leq \exp\left\{-\frac{n}{12} \log \frac{1}{1-q}\right\}, \quad \text{since } a > 1.
\end{aligned}$$

Hence, from (A.2.1), we have for any fixed  $\epsilon > 0$ ,

$$\begin{aligned}
&\mathbb{P}(G \text{ has no perfect matching}) \\
&\leq 2n(1-q)^n \cdot \left(1 + \left(\lceil \frac{n}{2} \rceil - 1\right) \exp\left\{-\frac{n}{12} \log \frac{1}{1-q}\right\}\right) \\
&\leq 2n(1-q)^n \cdot (1 + \epsilon), \quad \text{for } n \text{ large enough.} \tag{A.2.2}
\end{aligned}$$

Now, fix a node  $u_i \in \mathcal{U}$ . Let  $E_i$  denote the event that  $u_i$  is an isolated node. Then,  $\mathbb{P}(E_i) = (1-q)^n$ . It follows that

$$\mathbb{P}(G \text{ has no perfect matching}) \geq (1-q)^n.$$

Hence, putting  $\epsilon = 0.5$  in (A.2.2), we have (for large enough  $n$ )

$$(1-q)^n \leq \mathbb{P}(G \text{ has no perfect matching}) \leq 3n(1-q)^n. \tag{A.2.3}$$

This completes the proof.

### A.3 Proof of Lemma 2.2

Fix the number of queues (and servers),  $n$ , large enough for Theorem 2.1 to hold, and consider the evolution of  $Q_1$  under the above service rule.  $Q_1(t)$  evolves according to a Markov chain with the following state-transition probabilities:

$$\begin{aligned}
 p_0 &= \mathbb{P}(Q_1(t+1) = Q_1(t) + 1) \leq p \cdot 3n(1-q)^n, \\
 q_0 &= \mathbb{P}(Q_1(t+1) = Q_1(t) - 1 \geq 0) \geq (1-p)(1-3n(1-q)^n), \\
 \mathbb{P}(Q_1(t+1) = Q_1(t) + m) &= 0, \tag{A.3.1}
 \end{aligned}$$

for all  $m \notin \{0, 1, -1\}$ . Further, the evolution of  $Q_1$  is independent of the states of, and arrivals to, all the other queues. Figure A.1 shows the transition probabilities for  $Q_1(t)$ .

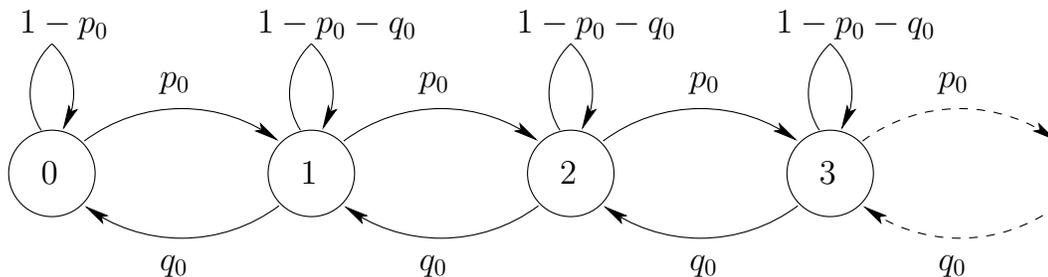


Figure A.1: Markov chain for the evolution of the first queue

For  $\rho := p_0/q_0 < 1$ , the steady-state distribution of the Markov chain in Figure A.1 is given by

$$\mathbb{P}(Q_1(t) = b) = (1 - \rho)\rho^b, \quad \forall b \geq 0,$$

implying  $\mathbb{P}(Q_1(t) > b) = \rho^{b+1}$ . Using (A.3.1), we get

$$\begin{aligned} \mathbb{P}(Q_1(t) > b) &\leq \left( \frac{3pn(1-q)^n}{(1-p)(1-3n(1-q)^n)} \right)^{b+1} \\ &\leq \left( \frac{6pn(1-q)^n}{1-p} \right)^{b+1}, \end{aligned}$$

for  $n$  large enough. The same calculation applies to each one of the queues from  $Q_2$  to  $Q_n$ , since the number of packets served from a queue  $Q_i$  is independent of all other queues and their respective arrivals; it is a function of the random variables  $X_{jk}(t)$ ,  $Q_i(t-1)$  and  $A_i(t)$ . Therefore, for the service rule under consideration,

$$\begin{aligned} \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) &\leq \sum_{i=1}^n \mathbb{P}(Q_i(0) > b) \\ &= n\mathbb{P}(Q_1(0) > b) \\ &\leq n \left( \frac{6pn(1-q)^n}{1-p} \right)^{b+1}. \end{aligned}$$

Hence,

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) \geq (b+1) \log \frac{1}{1-q},$$

which combined with (2.2.1), proves that the service rule under consideration maximizes (2.1.3).

## A.4 Proof of Lemma 2.3

Consider the evolution of  $Q_1$ , starting from any state  $Q_1(t)$ . The following event leads to  $\{Q_1(t+b+1) > b\}$ , irrespective of the channel realizations: for  $b+1$  consecutive timeslots  $(t+1, \dots, t+b+1)$ , there are arrivals to  $Q_1$ .

This event leads to  $Q_1(t + b + 1) > b$ , since in a given timeslot, at most 1 packet can be served from any given queue. The probability of this event is  $r^{b+1}$ . Therefore, under the perfect matching scheduling rule,

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) = 0,$$

taking steps similar to that in the proof of Theorem 2.1.

## A.5 Proof of Theorem 2.3

Under any algorithm in the iLQF class, the queue-length vector

$$\underline{Q}(t) := [Q_1(t), Q_2(t), \dots, Q_n(t)]^T$$

forms a Markov chain on a countable state space,  $\mathbb{W}^n$ , where  $\mathbb{W} = \{0, 1, 2, \dots\}$ . This is because the algorithm takes decisions based only upon the current queue-lengths and arrivals and channel states. We consider a new Markov chain

$$\underline{Z}(t) := [\underline{Q}(t), \underline{Q}(t-1), \dots, \underline{Q}(t-k_0+1)],$$

where we recall that  $k_0$  is the parameter in the *Drain* property.

$\underline{Z}(t)$  is a Markov chain under the given algorithm. We sample  $\underline{Z}(t)$  every  $k_0^{\text{th}}$  time-slot to get a Markov chain

$$\underline{B}(t) := \underline{Z}(k_0 t).$$

The Markov chains  $\underline{B}(t)$  and  $\underline{Z}(t)$  have the same stationary distribution. Let  $\underline{B}(t) = [B_{ij}(t)]$  with  $1 \leq i \leq n$  and  $1 \leq j \leq k_0$ . Define  $B^*(t) :=$

$\max_{1 \leq i \leq n} B_{i1}(t) = \max_{1 \leq i \leq n} Q_i(k_0 t)$ . Let  $\alpha_n = 3n(1 - q)^n$ . Then, for all  $m$ , we have:

$$\begin{aligned}
\mathbb{P}(B^*(t+1) < m | B^*(t) = m > 0) &\geq \frac{1}{2} && \text{(A.5.1)} \\
\mathbb{P}(B^*(t+1) = m+1 | B^*(t) = m) &\leq k_0 \alpha_n \\
\mathbb{P}(B^*(t+1) = m+2 | B^*(t) = m) &\leq \binom{k_0}{2} \alpha_n^2 \\
&\vdots \\
\mathbb{P}(B^*(t+1) = m+r | B^*(t) = m) &\leq \binom{k_0}{r} \alpha_n^r \\
&\vdots \\
\mathbb{P}(B^*(t+1) = m+k_0 | B^*(t) = m) &\leq \alpha_n^{k_0} \\
\mathbb{P}(B^*(t+1) > m+k_0 | B^*(t) = m) &= 0. && \text{(A.5.2)}
\end{aligned}$$

The inequality (A.5.1) follows from the Drain property in the statement of the theorem, while the transition probability bounds in (A.5.2) follow from union bound and Lemma 2.4.

Recall that the dominance property allows us to add packets to a queuing system and the resulting tail probabilities are only larger than without the packet additions. This motivates us to construct a queuing system  $\underline{R}$  where the packets are “carefully” added to ensure that the bounds in (A.5.1) and (A.5.2) are met with equality, as explained in the following.

We consider a queuing system  $\underline{R}$  that has the same sample-path wise external arrivals and channel realizations as the system  $\underline{Q}$ , and we add extra packets to one of the longest queues in  $\underline{R}$  at timeslots that are integer multiples

of  $k_0$ . Define

$$\tilde{\underline{Z}}(t) := [\underline{R}(t), \underline{R}(t-1), \dots, \underline{R}(t-k_0+1)],$$

$$\tilde{\underline{B}}(t) := \tilde{\underline{Z}}(k_0 t),$$

and

$$\tilde{B}^*(t) = \max_{1 \leq i \leq n} \tilde{B}_{i1}(t).$$

We want to ensure that the inequalities (A.5.1) and (A.5.2) are met with equality if we replace  $B^*(t)$  with  $\tilde{B}^*(t)$ , which will enable us to get bounds on  $\mathbb{P}(\tilde{B}^*(t) > b)$ , which will ultimately yield bounds on  $\mathbb{P}(B^*(t) > b)$ . To this end, define  $t' := k_0(t+1) - 1$ , and for  $j$  such that  $-\tilde{B}^*(t) \leq j \leq k_0$ , consider the sets of sample paths

$$\mathcal{C}_j := \left\{ \omega \in \Omega : \tilde{B}^*(t)(\omega) + j = \max_{1 \leq i \leq n} \left( R_i(t') + A_i(t') - \sum_{\ell} X_{i\ell}(t') Y_{i\ell}(t') \right)^+ (\omega) \right\},$$

where  $\Omega$  is the space over which all the random variables are defined. Each  $\omega \in \Omega$  belongs to precisely one of the sets  $\mathcal{C}_j$ . The set  $\mathcal{C}_j$  is the set of all sample paths where the maximum queue length changes by  $j$  over the  $k_0$  timeslots prior to and including the timeslot  $t'$ . This follows from the fact that in  $k_0$  timeslots, the maximum queue-length can increase by an amount at most  $k_0$ , and can possibly decrease to 0, therefore  $\mathbb{P}(\mathcal{C}_j)$  can possibly be nonzero only for  $-\tilde{B}^*(t) \leq j \leq k_0$ . Note that

$$\left( R_i(t') + A_i(t') - \sum_{\ell} X_{i\ell}(t') Y_{i\ell}(t') \right)^+ (\omega)$$

is the queue-length of the queue  $R_i$  at the end of service in timeslot  $t' + 1 = k_0(t+1)$ , but before extra packets are added. The drain property therefore

implies that

$$\mathbb{P}(\mathcal{C}_{-1}) + \mathbb{P}(\mathcal{C}_{-2}) + \cdots \geq \frac{1}{2}.$$

**Matching the Downcrossing Probabilities:**

Let  $j_0 := \min\{j : \sum_{i=1}^j \mathbb{P}(\mathcal{C}_{-i}) \geq 0.5\}$ . Such a  $j_0$  exists following the explanation above.

1. If  $\omega \in \mathcal{C}_{-j}$  with  $0 \leq j < j_0$ , then add  $j$  packets to the smallest-indexed element of the set  $\Xi$ , defined to be the set

$$\Xi := \arg \max_{1 \leq i \leq n} \left\{ \left( R_i(t') + A_i(t') - \sum_{\ell} X_{i\ell}(t') Y_{i\ell}(t') \right)^+ \right\}.$$

2. If  $\omega \in \mathcal{C}_{-j}$  with  $j > j_0$ , then add  $j - 1$  packets to the smallest-indexed element of the set  $\Xi$  after the service in  $k_0(t + 1)^{th}$  timeslot is complete.
3. If  $\omega \in \mathcal{C}_{-j_0}$ , then add  $j - 1 + D$  packets to the smallest-indexed element of the set  $\Xi$  after the service in  $k_0(t + 1)^{th}$  timeslot is complete, where the random variable  $D$  is a Bernoulli random variable, independent of all other random variables, with

$$\mathbb{P}(D = 0) = \frac{0.5 - \sum_{i=0}^{j_0-1} \mathbb{P}(\mathcal{C}_{-i})}{\mathbb{P}(\mathcal{C}_{-j_0})}.$$

Therefore, at this stage, if no further extra packets are added to the system  $\underline{R}$ , then we have (A.5.1) met with equality by  $\tilde{B}^*(t)$ , and the inequalities (A.5.2) are obeyed by  $\tilde{B}^*(t)$ .

**Matching the Upcrossing Probabilities:** Similarly, we can add packets to the system  $\underline{R}$  in the timeslot  $k_0(t + 1)$  to ensure that *both* the inequalities (A.5.1)

and (A.5.2) are met with equality. We skip details for the sake of brevity. Finally, we know from Kolmogorov's extension theorem ([8], Appendix A) that there exists a well-defined probability space which supports all the random variables necessary for the construction.

Thus, for the Markov chain  $\underline{\tilde{B}}$ ,  $\tilde{B}^*(t)$  obeys the inequalities in (A.5.2) with equality. Let the first column in the matrix  $\underline{B}(t)$  be denoted by  $B_1(t)$ , and similarly for  $\underline{\tilde{B}}(t)$ . Since  $\tilde{B}_1(t) = \underline{R}(k_0t)$  and  $B_1(t) = \underline{Q}(k_0t)$ , we have for all  $b \geq 0$ ,

$$\mathbb{P}\left(\max_{1 \leq i \leq n} R_i(0) > b\right) = \mathbb{P}(\tilde{B}^*(0) > b) \geq \mathbb{P}(B^*(0) > b) = \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right).$$

We note that the Markov chain  $\underline{\tilde{B}}(t)$  is positive recurrent for  $n$  large enough, because it is aperiodic, irreducible and the Lyapunov function for any valid state  $\underline{\tilde{B}}$  of the Markov chain is  $Lyap(\underline{\tilde{B}}) = \tilde{B}^*$  (the reason this works is that the maximum queue-length decreases by one with probability half, and increases by a finite amount with an arbitrarily small probability when  $n$  is large – thus, there is a negative drift whenever the value of the Lyapunov function is strictly positive, and Foster's theorem completes the proof).

Computing  $\mathbb{P}(\tilde{B}^*(0) > b)$  in the Steady State:

Partition  $\mathbb{W}^{k_0n}$  into the following disjoint sets, for  $j \geq 0$  :

$$V_j := \left\{ \left[ \begin{array}{cccc} x_{11} & x_{21} & \dots & x_{k_01} \\ x_{12} & x_{22} & \dots & x_{k_02} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1n} & x_{2n} & \dots & x_{k_0n} \end{array} \right] \in \mathbb{W}^{k_0n} : \max_{1 \leq i \leq n} x_{1i} = j \right\}.$$

Let  $\pi_j := \mathbb{P}(\underline{\tilde{B}}(0) \in V_j)$ , i.e.,  $\pi_j = \mathbb{P}(\tilde{B}^*(0) = j)$ .

Let  $V_j = \{v_{j_1}, v_{j_2}, \dots\}$ ,  $\sigma(j_i) = \mathbb{P}(\tilde{\underline{B}}(0) = v_{j_i})$  in the steady state, and let  $p(i_1, j_2)$  denote the probability of transition from state  $v_{i_1}$  to state  $v_{j_2}$  for the Markov chain  $\tilde{\underline{B}}(t)$ . Further, define  $\mathcal{L}_i := \{i+1, (i-1)^+, (i-2)^+, \dots, (i-k_0)^+\}$ . Then, the following equations hold by flow balance in the Markov chain  $\tilde{\underline{B}}(t)$  :

$$\begin{aligned}
\sum_{i=1}^{\infty} \sum_{j=1}^{k_0} \sum_{r=1}^{\infty} \sigma(0_i) p(0_i, j_r) &= \sum_{i=1}^{\infty} \sum_{r=1}^{\infty} \sigma(1_r) p(1_r, 0_i) \\
\sum_{i=1}^{\infty} \sum_{j=2}^{k_0+1} \sum_{r=1}^{\infty} \sigma(1_i) p(1_i, j_r) &= \sum_{s \in \mathcal{L}_1} \sum_{r=1}^{\infty} \sum_{i=1}^{\infty} \sigma(s_r) p(s_r, 1_i) \\
&\vdots \\
\sum_{i=1}^{\infty} \sum_{j=y+1}^{y+k_0} \sum_{r=1}^{\infty} \sigma(y_i) p(y_i, j_r) &= \sum_{s \in \mathcal{L}_y} \sum_{r=1}^{\infty} \sum_{i=1}^{\infty} \sigma(s_r) p(s_r, y_i) \\
&\vdots
\end{aligned} \tag{A.5.3}$$

The equations (A.5.3) can be rewritten as:

$$\sum_{j=y+1}^{y+k_0} \mathbb{P}(\tilde{\underline{B}}(0) \in V_y, \tilde{\underline{B}}(1) \in V_j) = \sum_{s \in \mathcal{L}_y} \mathbb{P}(\tilde{\underline{B}}(0) \in V_s, \tilde{\underline{B}}(1) \in V_y)$$

or, as:

$$\pi_y \sum_{j=y+1}^{y+k_0} \mathbb{P}(\tilde{B}^*(1) = j | \tilde{B}^*(0) = y) = \sum_{s \in \mathcal{L}_y} \pi_s \mathbb{P}(\tilde{B}^*(1) = y | \tilde{B}^*(0) = s). \tag{A.5.4}$$

The conditional probabilities in (A.5.4) are given by the right-hand-sides in (A.5.2), since (by construction) the Markov chain  $\tilde{\underline{B}}$  obeys these bounds with equality.

We consider  $n$  large enough, so that  $\alpha_n = 3n(1-q)^n \leq 1$ . For  $m \geq 0$ , we prove the following statement about  $\pi_m$  by induction:

$$g(m) : \pi_m \leq \pi_0 \cdot 5^{k_0 m} \alpha_n^m.$$

Base Case:

Clearly,  $g(0)$  is true, since  $\pi_0 = \pi_0$ .

Induction Step:

Let  $g(0), g(1), \dots, g(m-1)$  be true, and we need to prove  $g(m)$ . From Equations (A.5.2) and (A.5.4), and noting that  $\pi_j = 0$  for  $j < 0$ , we have:

$$\frac{\pi_m}{2} = \pi_{m-1} \sum_{j=1}^{k_0} \binom{k_0}{j} \alpha_n^j + \pi_{m-2} \sum_{j=2}^{k_0} \binom{k_0}{j} \alpha_n^j + \dots + \pi_{m-k_0} \alpha_n^{k_0}.$$

Thus,

$$\begin{aligned} \pi_m &= 2 \sum_{r=1}^{k_0} \left( \pi_{m-r} \sum_{j=r}^{k_0} \binom{k_0}{j} \alpha_n^j \right) \\ &\leq 2 \sum_{r=1}^{k_0} \left( \pi_{m-r} \alpha_n^r \sum_{j=r}^{k_0} \binom{k_0}{j} \right) \\ &\leq 2 \sum_{r=1}^{k_0} (\pi_{m-r} \alpha_n^r 2^{k_0}) \\ &\stackrel{(a)}{\leq} 2 \sum_{r=1}^{k_0} (\pi_0 \cdot 5^{k_0(m-r)} \alpha_n^{m-r} \alpha_n^r 2^{k_0}) \\ &= 2^{k_0+1} \alpha_n^m \cdot \pi_0 \sum_{r=1}^{k_0} 5^{k_0(m-r)} \\ &= \pi_0 \alpha_n^m 2^{k_0+1} 5^{k_0(m-k_0)} \frac{5^{k_0^2} - 1}{5^{k_0} - 1} \\ &\leq \pi_0 \alpha_n^m 5^{k_0 m} \frac{2^{k_0+1}}{5^{k_0} - 1} \\ &\leq \pi_0 \alpha_n^m 5^{k_0 m}, \end{aligned}$$

since  $2^{k+1} \leq 5^k - 1$  for all  $k \geq 1$ . Here, the inequality (a) follows from induction

hypothesis. Hence, by principle of mathematical induction,  $g(m)$  is true for all integers  $m \geq 0$ .

Now,

$$\sum_{m=1}^{\infty} 5^{k_0 m} [3n(1-q)^n]^m = \frac{5^{k_0} [3n(1-q)^n]}{1 - 5^{k_0} [3n(1-q)^n]} \xrightarrow{n \rightarrow \infty} 0.$$

Since  $\pi_0 + \pi_1 + \cdots = 1$ , we have

$$\pi_0 = \frac{1}{1 + \sum_{m=1}^{\infty} \frac{\pi_m}{\pi_0}} \geq \frac{1}{1 + \sum_{m=1}^{\infty} 5^{k_0 m} [3n(1-q)^n]^m} > \frac{1}{3},$$

for  $n$  large enough. Further,

$$\begin{aligned} \mathbb{P}(\tilde{B}^*(0) > b) &= \sum_{m=b+1}^{\infty} \pi_m = \pi_0 \sum_{m=b+1}^{\infty} \frac{\pi_m}{\pi_0} \\ &\leq \pi_0 5^{k_0(b+1)} [3n(1-q)^n]^{b+1} \sum_{m=0}^{\infty} 5^{k_0 m} [3n(1-q)^n]^m \\ &= \pi_0 5^{k_0(b+1)} [3n(1-q)^n]^{b+1} \frac{1}{1 - 5^{k_0} [3n(1-q)^n]} \\ &\leq 2\pi_0 5^{k_0(b+1)} [3n(1-q)^n]^{b+1}, \end{aligned}$$

for  $n$  large enough. Hence,

$$\begin{aligned} \limsup_{n \rightarrow \infty} \frac{1}{n} \log \mathbb{P}(\tilde{B}^*(0) > b) &\leq \limsup_{n \rightarrow \infty} \frac{1}{n} [\log 2 + \log \pi_0 + k_0(b+1) \log 5 \\ &\quad + (b+1) \log(3n) + n(b+1) \log(1-q)] \end{aligned}$$

Noting that  $\limsup_n (a_n + b_n) \leq \limsup_n a_n + \limsup_n b_n$  and that  $\limsup_{n \rightarrow \infty} \frac{1}{n} \log \pi_0 = 0$  since  $\pi_0 > \frac{1}{3}$  for  $n$  large enough, we get

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}(\tilde{B}^*(0) > b) \geq (b+1) \log \frac{1}{1-q},$$

and, by (2.2.1), the proof of Theorem 2.3 is complete.

## A.6 Proof of Lemma 2.5

We prove that if  $\mathcal{M}_k$  is a matching, then so is  $\mathcal{M}_{k+1}$ , and  $|\mathcal{M}_k| = |\mathcal{M}_{k+1}|$ . We need to focus only on the case where in step 2 of PullUp, the node  $v_k$  has no incoming edge and there exists  $v_l \in \Delta(G_k, v_k)$  with  $l > k$ . Let the path from  $v_k$  to  $v_l$  be  $v_k \rightarrow u_{i_1} \rightarrow v_{j_1} \rightarrow u_{i_2} \rightarrow v_{j_2} \cdots \rightarrow u_{i_c} \rightarrow v_l$ . Let

$$\mathcal{M}_k = \{(u_{i_1}, v_{j_1}), (u_{i_2}, v_{j_2}), \dots, (u_{i_x}, v_{j_x})\},$$

with  $v_{j_c} = v_l$ , and  $v_{j_y} \neq v_k$  for any  $y$ . Then, by definition,

$$\mathcal{M}_{k+1} = \{(u_{i_1}, v_k), (u_{i_2}, v_{j_1}), \dots, (u_{i_c}, v_{i_{c-1}}), (u_{i_{c+1}}, v_{j_{c+1}}), \dots, (u_{i_x}, v_{j_x})\}.$$

Hence,  $|\mathcal{M}_{k+1}| = |\mathcal{M}_k|$ . Further, the edges in  $\mathcal{M}_{k+1}$  are node-disjoint because all the nodes  $v_k, v_{i_1}, v_{i_2}, \dots, v_{i_x}$  are different. Hence,  $\mathcal{M}_{k+1}$  is a matching. Therefore,  $\mathcal{M}' = \text{PullUp}(G, \mathcal{M}, \mathcal{V})$  is a matching and  $|\mathcal{M}'| = |\mathcal{M}|$ .

## A.7 Proof of Lemma 2.6

Consider a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$  with a given matching  $\mathcal{M}$ ,  $|\mathcal{U}| = |\mathcal{V}| = n$  and  $|\mathcal{E}| = m$ .

1. For each node  $v_k \in \mathcal{V}$ , the set of all nodes reachable from  $v_k$  can be found in  $O(m + n)$  computations via Depth First Search (DFS) (Theorem 3.13 in [19]). Since  $m = O(n^2)$  and there are at most  $n$  nodes from which the set of reachable nodes needs to be found out, the operation  $\text{PullUp}(G, \mathcal{M}, \mathcal{V})$  can be completed in  $O(n^3)$  operations.

2. For a matching  $\mathcal{M} = \{(u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), \dots, (u_{a_k}, v_{b_k})\}$ , define

$$SUM(\mathcal{M}) := \sum_{i=1}^k (a_i + b_i).$$

Let  $PullUp(G, \mathcal{M}, \mathcal{V}) = \mathcal{M}'$ . If  $\mathcal{M} \neq \mathcal{M}'$ , then  $SUM(\mathcal{M}) \leq SUM(\mathcal{M}') - 1$ . Since  $SUM(\mathcal{M}_1)$  is  $O(n^2)$ , the number of times the PullUp operation is performed in the step 2b of the algorithm is  $O(n^2)$ .

The step 1 of the algorithm can be implemented in  $O(n)$  computations. For step 2, the set  $\mathcal{Q}_L$  can be found in  $O(n)$  computations. From [16], we know that in a bipartite graph  $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$  with  $|\mathcal{U}| = |\mathcal{V}| = n$  and  $|\mathcal{E}| = m$ , it is possible to find a largest cardinality matching in  $O(m\sqrt{n})$  computations. Adding isolated dummy nodes if necessary, the graph  $G_L$  can be made to have  $n$  nodes on either side. Further,  $m = O(n^2)$ . A largest matching can therefore be found with  $O(n^{2.5})$  computations. For any fixed server node, the set of all nodes to which this server node has a path can be found by depth-first search (DFS) in  $O(m + n)$  computations ([19], Thm. 3.13) and since  $m = O(n^2)$ , in  $O(n^2)$  computations. The step 2a can thus be performed in  $O(n^3)$  computations, while the step 2b can be performed in  $O(n^4)$  computations. Noting that the step 2b needs to be executed at most once, every round (except possibly the last round) can be implemented in  $O(n^3)$  computations, and the last round can be performed in  $O(n^4)$  computations.

By step 3, the number of rounds to be completed is at most  $L$ . However, if  $\mathcal{Q}_L = \emptyset$  for a round, then no computations need to be performed for that

round at all, since  $\mathcal{M}_L$  is the vacuous matching of cardinality 0. Hence, instead of redefining  $L$  to  $L-1$  in step 3, we can define  $L$  to be the length of the longest queue at the end of that round, and the algorithm will result in the same set of allocations of queues to servers as before. The maximum of the queue-lengths can be obtained in  $O(n)$  computations. With this modification, the number of rounds is at most  $n$ , since every round allocates at least one server, or else it is the last round.

Thus, all the perfect matching rounds (i.e., except possibly the last round of maximal matching) can be implemented in  $O(n^4)$  computations, and the last round of largest matching (step 2b) can be implemented in  $O(n^4)$  computations. Finally, the output (updated queue-lengths and server allocation decisions) can be reported in  $O(n)$  computations (memory reads), since at most  $n$  of the  $Y_{ij}(t)$  are nonzero.

Therefore, the proposed algorithm can be implemented in  $O(n^4)$  computations per timeslot.

## A.8 Proof of Lemma 2.7

Consider  $b > a$ . If the node  $v_a$  has an incoming edge in the graph  $G_b$ , then it has an incoming edge in  $G_{b+1}$ , even if  $G_b \neq G_{b+1}$ . This is because if  $G_b \neq G_{b+1}$ , then the node  $v_b$  does not have an incoming edge in  $G_b$  and there exists a path from  $v_b$  to  $v_c$ ,  $c > b$  in  $G_b$ . Even if this path contains  $v_a$ , the incoming edge to  $v_a$  (in  $G_b$ ) becomes an outgoing edge, while another one of  $v_a$ 's outgoing edges, on reversal, becomes an incoming edge, since the path

cannot terminate on  $v_a$ . Hence, if  $v_a$  has an incoming edge in  $G_{a+1}$ , then it has an incoming edge in  $G_{n+1}$ .

Now, let  $v_a \in \mathcal{V}$  and assume that  $v_a$  has no incoming edge in  $G_{n+1}$ , therefore in  $G_{a+1}$ , implying  $G_a = G_{a+1}$ . The following notation is used throughout this proof:

$$\begin{aligned}
\mathcal{N}_a(b) &= \text{The set of all nodes reachable} \\
&\quad \text{from } v_a \text{ in the graph } G_b \\
\Gamma(G_b, v_a) &= \mathcal{N}_a(b) \cap \mathcal{U} \\
\Delta(G_b, v_a) &= \mathcal{N}_a(b) \cap \mathcal{V} \\
\mathcal{E}_a &= \text{The set of edges in the graph } G_a
\end{aligned}$$

(According to this notation,  $\mathcal{N}_a = \mathcal{N}_a(a)$ .) Thus, we have  $\Delta(G_a, v_a) \subseteq \{v_1, v_2, \dots, v_{a-1}\}$ ,

and  $\Gamma(G_a, v_a) = \mathcal{S}_1 \cup \mathcal{S}_2$ , where

$$\begin{aligned}
\mathcal{S}_1 &= \{u_i : (u_i, v_j) \in \mathcal{M}_a \text{ for some } v_j \in \Delta(G_a, v_a)\}, \\
\mathcal{S}_2 &= \{u_i : (v_j, u_i) \in \mathcal{E}_a \text{ for some } v_j \in \Delta(G_a, v_a), \\
&\quad \text{and } u_i \text{ has no outgoing edge}\}.
\end{aligned}$$

To see this, note that any node  $v_j \in \Delta(G_b, v_a)$  has exactly one incoming (forward) edge, since the forward edges belong to a matching. Therefore,  $\mathcal{S}_1 \cup \mathcal{S}_2 \subseteq \Gamma(G_a, v_a)$ . Further, every node  $u_i \in \Gamma(G_a, v_a)$  is reachable from  $v_a$ , so if  $u_i$  has an outgoing (forward) edge, it must, by definition, end on some  $v_j \in \Delta(G_a, v_a)$ , or else it must not have an outgoing edge. Hence,  $\Gamma(G_a, v_a) = \mathcal{S}_1 \cup \mathcal{S}_2$ .

We now prove the following statement, which immediately implies the claim: for all  $b > a$ , if  $v_b$  has no incoming edge in  $G_b$ , and there exists a path from  $v_b$  to  $v_c$  (with  $c > b$ ) in  $G_b$ , then that path does not contain any node from  $\mathcal{N}_a$ . (★)

Before proving the statement  $(\star)$ , let us see how it implies the claim that in the graph  $G_{n+1}$ , there exists no directed path from  $v_a$  to any node  $v_b$  for  $b > a$ . Fix any  $b > a$ . If  $v_b$  has an incoming edge, then we have  $G_b = G_{b+1}$ . If  $v_b$  has no incoming edge, and in the graph  $G_b$  there exists no path from  $v_b$  to  $v_c$  for any  $c > b$ , then again  $G_b = G_{b+1}$ . If  $v_b$  has no incoming edge in  $G_b$  and there exists a path from  $v_b$  to  $v_c$  with  $c > b$ , then (by  $(\star)$ ) this path contains no node from  $\mathcal{N}_a$ . Thus, in every possible scenario, the edge-configuration (i.e., the (directed and labeled) pattern of incoming and outgoing edges) for the nodes in  $\mathcal{N}_a$  is the same in both the graphs  $G_b$  and  $G_{b+1}$ . Thus, if the node  $v_a$  has no incoming edge in the graph  $G_{a+1}$ , then it does not have an incoming edge in the graph  $G_{n+1}$ , proving the first part of the claim.

Now, if  $\text{PullUp}(G, \mathcal{M}', \mathcal{V}) \neq \mathcal{M}'$ , then the following is true: if all the edges in  $G$  that belong to  $\mathcal{M}'$  are forward edges and all the other edges are backward, then there exists a node  $v_a$  with no incoming edges, and has a directed path to a node  $v_b$ , with  $b > a$ . No such path exists by the previous argument, completing the proof if  $(\star)$  is true.

**Proof of  $(\star)$ :**

If the statement  $(\star)$  is true, then the set of nodes in  $\mathcal{V}$  reachable from  $v_a$  under  $G_a$  is the same as those under  $G_b$  for any  $b > a$ , in particular for  $b = n + 1$ . Suppose, for obtaining a contradiction, that the statement  $(\star)$  is false, and let  $b$  be the smallest index greater than  $a$  for which the statement is false. Therefore,  $\mathcal{N}_a(a) = \mathcal{N}_a(b)$ , since the edge-configurations for all the nodes in  $\mathcal{N}_a(a)$  are unchanged under the transformations that convert the graph  $G_a$

to  $G_b$ , by definition of  $b$ . Let the path from  $v_b$  to a node  $v_c$ ,  $c > b$  contain one or more nodes from  $\mathcal{N}_a(a)$ . Let this path be  $v_b \rightarrow u_{i_1} \rightarrow v_{j_1} \rightarrow u_{i_2} \rightarrow v_{j_2} \rightarrow \dots \rightarrow v_{j_k} = v_c$ . This path does not contain any node from  $\mathcal{S}_2$ , since the nodes in  $\mathcal{S}_2$  have no outgoing edges. If the path contains a node  $u_i \in \mathcal{S}_1$ , then it contains the node  $v_j$  for which  $(u_i, v_j) \in \mathcal{E}_b$ , the *only* forward edge associated with  $u_i$ . Since the edge configurations of the nodes in  $\mathcal{N}_a(a) = \mathcal{N}_a(b)$  are the same under the graphs  $G_a$  and  $G_b$ , we have  $(u_i, v_j) \in \mathcal{E}_a$ . Hence, the path from  $v_b$  to  $v_c$  contains at least one node from  $\Delta(G_a, v_a) = \Delta(G_b, v_a)$ .

Define

$$j_0 := \min\{j : v_j \notin \mathcal{N}_a(b) \forall j \geq j_0\}.$$

Since  $v_c = v_{j_k} \notin \mathcal{N}_a(b)$ , we have  $0 < j_0 \leq k$ . Then,  $v_{j_0-1} \in \mathcal{N}_a(a)$ , and the edge  $v_{j_0-1} \rightarrow u_{j_0}$  is present in  $G_a$ , since the edge configuration of  $v_{j_0-1}$  is the same under  $G_a$  and  $G_b$ . Hence,  $u_{j_0} \in \mathcal{N}_a(a)$ , and the edge configuration of  $u_{j_0}$  is the same under  $G_a$  and  $G_b$ , implying  $v_{j_0} \in \mathcal{N}_a(a)$ , a contradiction to the definition of  $j_0$  since  $\mathcal{N}_a(a) = \mathcal{N}_a(b)$ . Therefore, the statement  $(\star)$  is true.

## A.9 Proof of Lemma 2.8

The following notation is used throughout this proof:

- $\mathcal{M}_r$  = The set of queues served in the  $r^{th}$  round, in the system  $\underline{R}$
- $\mathcal{Y}_r$  = The set of servers allocated in the  $r^{th}$  round, in the system  $\underline{R}$
- $\mathcal{N}_r$  = The set of queues served in the  $r^{th}$  round, in the system  $\underline{Q}$
- $\mathcal{Z}_r$  = The set of servers allocated in the  $r^{th}$  round, in the system  $\underline{Q}$

For simplicity of notation, throughout this proof, we use  $Q_i^{(r)}$  and  $R_i^{(r)}$  to denote  $Q_i^{(r)}(t)$  and  $R_i^{(r)}(t)$  respectively. By definition,  $R_i^{(0)} := R_i(t-1) + A_i(t)$  and

$Q_i^{(0)} := Q_i(t-1) + A_i(t)$ . Let  $\hat{R} := \max_i R_i^{(0)}$ ,  $\hat{Q} := \max_i Q_i^{(0)}$  and  $w := \hat{R} - \hat{Q}$ . Let there exist  $n_R$  and  $n_Q$  rounds of perfect matchings in the system  $\underline{R}$  and  $\underline{Q}$  respectively.

Case 1:  $n_R < w$ .

If a queue  $R_i$  was served even once in the  $n_R$  rounds, then at the end of  $n_R$  rounds,  $R_i^{(n_R)} = \hat{R} - n_R > \hat{R} - w = \hat{Q}$ . Since there are exactly  $n_R$  rounds of perfect matching in the system  $\underline{R}$ ,

$$R_i(t) \geq R_i^{(n_R)} - 1 \geq \hat{Q} \geq Q_i(t).$$

If  $R_i$  was not served even once in the first  $n_R$  rounds of perfect matching, but was served in the last round of maximal matching, then

$$R_i(t) = \hat{R} - (n_R + 1) \geq \hat{R} - w = \hat{Q} \geq Q_i(t).$$

Finally, if the queue  $R_i$  was not served at all, then

$$R_i(t) = R_i(t-1) + A_i(t) \geq Q_i(t-1) + A_i(t) \geq Q_i(t),$$

and the claim is true in this case.

Case 2:  $n_R = w$ .

We have  $R_i^{(n_R)} \geq \hat{R} - n_R = \hat{Q}$ , with equality holding if and only if  $R_i^{(0)} \geq \hat{Q}$ . Let  $\mathcal{R}_{last} = \{R_{i_1}, R_{i_2}, \dots, R_{i_a}\}$  denote the set of longest (i.e. of length  $\hat{Q}$ ) queues at the beginning of the maximal matching round for the system  $\underline{R}$ , with  $i_1 < i_2 < \dots < i_a$ . Let  $\mathcal{Q}_{first} = \{Q_{j_1}, Q_{j_2}, \dots, Q_{j_b}\}$  denote the set of longest queues in the system  $\underline{Q}$ , at the beginning of the first round, with

$j_1 < j_2 < \dots < j_b$ . Then,  $\{j_1, j_2, \dots, j_b\} \subseteq \{i_1, i_2, \dots, i_a\}$ . If the first round in the system  $\underline{Q}$  is a perfect matching round (i.e.  $n_Q > 0$ ), then all of the queues in  $\underline{Q}_{first}$  are served, and only some of  $\underline{R}_{last}$ , and the claim is true because the queues in the system  $\underline{R}$  are not served for more than  $n_R + 1$  rounds.

Now, let  $n_Q = 0$ . Let a queue  $R_{i_c}$  be served by a server  $S_a$  in the  $(n_R + 1)^{th}$  round, but  $Q_{i_c}$  is not served in the 1<sup>st</sup> (largest matching) round. Then,  $S_a$  must serve a queue  $Q_{i_d}$  with  $d < c$ , otherwise the size of the largest matching can be strictly increased ( $\because X_{i_c a} = 1$ ), or there exists a directed path  $Q_{i_c} \rightarrow S_a \rightarrow Q_{i_d}$ , contradicting Lemma 2.7. The queue  $R_{i_d}$  must be served by a server  $S_e$ , otherwise there exists a directed path  $R_{i_d} \rightarrow S_d \rightarrow R_{i_c}$ , again contradicting Lemma 2.7. The server  $S_e$  must serve a queue  $Q_{i_f}$  with  $f < c$ , otherwise the size of the largest matching in  $\underline{Q}$  can be strictly increased (by allocating  $S_e$  to  $Q_{i_d}$ ,  $S_a$  to  $Q_{i_c}$ ), or there exists a directed path  $Q_{i_c} \rightarrow S_a \rightarrow Q_{i_d} \rightarrow S_e \rightarrow Q_{i_f}$  and  $f > c$ , contradicting the specifications of the algorithm and in particular, Lemma 2.7. This process of finding newer servers and queues in the two systems can be continued indefinitely, contradicting the finiteness of the number of queues and servers in the system. Therefore, if a queue  $R_{i_c}$  is served in the largest matching round of the system  $\underline{R}$ , then so is  $Q_{i_c}$  in the system  $\underline{Q}$ , and the claim holds in this case.

Case 3:  $n_R > w$ .

We prove the following statement  $f(r)$ , for  $0 \leq r \leq n_R - w$ , by induction:

$$f(r) : \mathcal{N}_r \subseteq \bigcup_{i=1}^{r+w} \mathcal{M}_i, \mathcal{Z}_r \subseteq \bigcup_{i=1}^{r+w} \mathcal{Y}_i, \text{ and } Q_i^{(r)} \leq R_i^{(r+w)} \text{ for all } i.$$

**Base case:**

We need to prove that  $f(0)$  is true. Since  $\mathcal{N}_0 = \emptyset$  and  $\mathcal{Z}_0 = \emptyset$ , we only need to prove that  $Q_i^{(0)} \leq R_i^{(w)}$ . Now, if  $R_i$  was not served during the first  $w$  rounds, then  $R_i^{(w)} = R_i^{(0)} \geq R_i^{(0)}$ . If  $R_i$  was served in at least one of the first  $w$  rounds of service, then  $R_i^{(w)} \geq \hat{R} - w = \hat{Q} \geq Q_i^{(0)}$ . Hence,  $f(0)$  is true.

**Induction step:**

Suppose  $f(0), \dots, f(r-1)$  are true for some  $r \geq 1$ . We need to prove  $f(r)$ . Let  $R_i \in \mathcal{M}_{r+w}$ . We prove that if  $R_i^{(r-1+w)} = Q_i^{(r-1)}$ , then  $Q_i \in \mathcal{N}_r$ . Since  $R_i \in \mathcal{M}_{r+w}$ , it was, at the beginning of that round, a longest queue.

Let  $R_i \in \mathcal{M}_{r+w}$  be allocated a server  $S_a$  in the  $(r+w)^{th}$  round. Therefore,  $X_{ia} = 1$ . Since (by induction hypothesis)

$$\bigcup_{i=1}^{r-1} \mathcal{Z}_i \subseteq \bigcup_{i=1}^{r-1+w} \mathcal{Y}_i, \text{ and } S_a \notin \bigcup_{i=1}^{r-1+w} \mathcal{Y}_i,$$

we have

$$S_a \notin \bigcup_{i=1}^{r-1} \mathcal{Z}_i,$$

so the server  $S_a$  is available to serve  $Q_i$  in the  $r^{th}$  round. Therefore, if there exists a perfect matching in the system  $\underline{R}$  in the  $(r+w)^{th}$  round, then there exists a perfect matching in the  $r^{th}$  round in the system  $\underline{Q}$ , and  $Q_i \in \mathcal{N}_r$ , implying that  $Q_i^{(r)} \leq R_i^{(r+w)}$ .

Now, for the purpose of obtaining a contradiction, let  $S_c \in \mathcal{Z}_r$ , and  $S_c \notin \mathcal{Y}_1 \cup \dots \cup \mathcal{Y}_{r+w}$ . Let  $Q_i$  be served by  $S_c$  in the  $r^{th}$  round, while  $R_i$  was served by  $S_d$  in  $(r+w)^{th}$  round. Hence,  $d < c$ .  $S_d$  must serve some queue

$Q_e$  in the system  $\underline{Q}$  in  $r^{th}$  round, because otherwise it can replace  $S_c$  to serve  $Q_i$  and the server  $S_d$  was unused (in the system  $\underline{Q}$ ) until the beginning of the  $r^{th}$  round by induction hypothesis.  $R_e$ , in turn, must be served by a server  $S_f$  in the  $(r + w)^{th}$  round in the system  $\underline{R}$ . We must have  $f < c$ , otherwise there exists a connecting path  $S_c \rightarrow R_i \rightarrow S_d \rightarrow R_e \rightarrow S_f$  and  $S_c$  cannot remain unused in the system  $\underline{R}$ , according to Lemma 2.7. This process can be continued indefinitely, contradicting the fact that the number of queues and servers is finite. Hence,  $\mathcal{Z}_r \subseteq \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \dots \cup \mathcal{Y}_{r+w}$ , and the induction is complete.

Hence, if we compare the state of the system  $\underline{R}$  after  $n_R$  rounds of perfect matching (i.e. at the beginning of the maximal matching round) and  $\underline{Q}$  at the end of  $n_Q - w$  rounds of perfect matching, we have the following:

1. The set of unallocated servers available in the system  $\underline{Q}$  is a superset of the set of unallocated servers available in the system  $\underline{R}$ .
2. The set of longest queues in the system  $\underline{Q}$  is a subset of the set of longest queues in the system  $\underline{R}$ .

As before, let  $\mathcal{R}_{last} = \{R_{i_1}, R_{i_2}, \dots, R_{i_a}\}$  denote the set of longest queues at the beginning of the maximal matching round for the system  $\underline{R}$ , with  $i_1 < i_2 < \dots < i_a$ . Let  $\mathcal{Q}_{first} = \{Q_{j_1}, Q_{j_2}, \dots, Q_{j_b}\}$  denote the set of longest queues in the system  $\underline{Q}$ , at the beginning of the  $(n_R - w + 1)^{th}$  round, with  $j_1 < j_2 < \dots < j_b$ . Then,  $\{j_1, j_2, \dots, j_b\} \subseteq \{i_1, i_2, \dots, i_a\}$ . If the  $(n_R - w + 1)^{th}$  round in the system  $\underline{Q}$  is a perfect matching round (i.e.

$n_Q > n_R - w$ ), then all of the queues in  $\mathcal{Q}_{first}$  are served, and only some of  $\mathcal{R}_{last}$ , and the claim is true because the queues in the system  $\underline{R}$  are not served for more than  $n_R + 1$  rounds.

Now, let  $n_Q = n_R - w$ . We need to prove that if a queue  $R_i$  is served in the largest matching round of the system  $\underline{R}$ , then so is  $Q_i$  in the system  $\underline{Q}$ . The proof is almost identical to that of the case  $n_R = w$ , and is skipped to avoid repetition. Therefore, the proof of the theorem is complete.

## A.10 Proof of Lemma 2.9

Let

$$\begin{aligned}\mathcal{M} &= \{(u_{i_1}, v_{j_1}), (u_{i_2}, v_{j_2}), \dots, (u_{i_x}, v_{j_x})\}, \\ \mathcal{M}_\star &= \{(u_{i_1}, v_{k_1}), (u_{i_2}, v_{k_2}), \dots, (u_{i_x}, v_{k_x})\},\end{aligned}$$

with  $k_y \leq a$  for all  $y \in \{1, 2, \dots, x\}$ . For obtaining a contradiction, if possible, let there exist an edge  $(u_{i_c}, v_b) \in \mathcal{M}'$  with  $c \leq x$  and  $b > a$ . Then there exists a node  $v_{k_d}$ ,  $d \leq x$ , such that no edge in  $\mathcal{M}'$  has  $v_{k_d}$  as one of its endpoints. Let  $(u_{i_d}, v_{\alpha_1}) \in \mathcal{M}'$  for some  $\alpha_1 < k_d$  (by Lemma 2.7). Therefore, in the matching  $\mathcal{M}_\star$ , the node  $v_{\alpha_1}$  is an endpoint of some edge  $(u_{\beta_1}, v_{\alpha_1})$ , else there exists a directed path in  $G^\ddagger$  from  $v_{\alpha_1}$  to  $v_{k_d}$ , namely  $v_{\alpha_1} \rightarrow u_{i_d} \rightarrow v_{k_d}$ , contradicting property 3 of  $\mathcal{M}_\star$  as required by the statement of the Lemma. There must exist an edge  $(u_{\beta_1}, v_{\alpha_2}) \in \mathcal{M}'$ , with  $\alpha_2 < k_d$  by Lemma 2.7. Hence, the node  $v_{\alpha_2}$  is an endpoint of an edge in  $\mathcal{M}_\star$ , since there exists a directed path  $v_{\alpha_2} \rightarrow u_{\beta_1} \rightarrow v_{\alpha_1} \rightarrow u_{i_d} \rightarrow v_{k_d}$ . This process can be continued indefinitely, contradicting the

finiteness of the number of nodes in  $\mathcal{U} \cup \mathcal{V}$ . Hence, the proof is complete.

## A.11 Proof of Theorem 2.5

Fix any  $\epsilon \in (0, \alpha)$ . By (2.6.2), there exists a strictly increasing sequence of natural numbers  $\{n_1, n_2, \dots\}$  such that for all  $k \geq 1$ ,

$$\max_{1 \leq i \leq n_k} p_i^{(n_k)} \in (\alpha - \epsilon, \alpha + \epsilon).$$

Hence, in the system  $\Upsilon'_{n_k}$ , there exists a queue  $Q_{i_k}^{[n_k]}$  that has the packet-arrival probability at least  $\alpha - \epsilon$ . Consider the following event that, under *any* scheduling algorithm, implies  $\{Q_{i_k}^{[n_k]}(0) > b\}$ : for  $b + 1$  consecutive timeslots before (and including) timeslot 0, there are arrivals to  $Q_{i_k}^{[n_k]}$ , and all the channels connecting  $Q_{i_k}^{[n_k]}$  to the servers are OFF in each of the  $b + 1$  timeslots. The probability of this event is at least  $(\alpha - \epsilon)^{b+1}((1 - q)^{n_k})^{b+1}$ . Hence, for all  $k \geq 1$ ,

$$\mathbb{P} \left( \max_{1 \leq i \leq n_k} Q_i^{[n_k]}(0) > b \right) \geq (\alpha - \epsilon)^{b+1}((1 - q)^{n_k})^{b+1},$$

implying

$$\limsup_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \leq (b + 1) \log \frac{1}{1 - q},$$

and the result is proved.<sup>1</sup>

---

<sup>1</sup>We are using the following results:

1. For two sequences of real numbers  $\{a_n\}$  and  $\{b_n\}$  with  $a_n \geq b_n$  for infinitely many values of  $n$ , we have

$$\liminf_{n \rightarrow \infty} a_n \geq \liminf_{n \rightarrow \infty} b_n.$$

## A.12 Proof of Theorem 2.6

From (2.6.2), it follows that there exists a natural number  $n_0$  such that for all  $n \geq n_0$ ,

$$\max_{1 \leq i \leq n} p_i^{(n)} \leq \frac{1 + \alpha}{2} < 1.$$

Consider a sequence of systems  $\{\Upsilon_n''\}$ . The channel process of the system  $\Upsilon_n''$  is identical to that of the system  $\Upsilon_n'$  (and  $\Upsilon_n$ ), but the arrival process is given by  $A_i''^{[n]}(t) = A_i'^{[n]}(t)$  for  $n < n_0$ , and by

$$A_i''^{[n]}(t) = \begin{cases} 1 & \text{with probability } \frac{1+\alpha}{2}, \\ 0 & \text{with probability } 1 - \frac{1+\alpha}{2}, \end{cases}$$

for  $n \geq n_0$ . Further, let  $\{A_i'^{[n]}(t) = 1\} \Rightarrow \{A_i''^{[n]}(t) = 1\}$ . By Kolmogorov's extension theorem ([8], Appendix A), there exists a probability space on which the above construction is valid and well-defined. Let the two systems  $\Upsilon_n'$  and  $\Upsilon_n''$  be started with the same initial queue-lengths. Then, by the sample-path-wise dominance property of the iLQF with PullUp algorithm (Lemma 2.8), it follows that for the two systems,

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i'^{[n]}(0) > b \right) \leq \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i''^{[n]}(0) > b \right).$$

The sequence of systems  $\Upsilon_n''$ , for  $n \geq n_0$  is identical to the one considered in Section 2.1, in particular a symmetric arrival system. Hence, Theorem 2.4

---

2. For a sequence  $\{a_n\}$ , we have

$$\liminf_{n \rightarrow \infty} a_n = - \limsup_{n \rightarrow \infty} (-a_n).$$

implies

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \geq (b+1) \log \frac{1}{1-q}.$$

The result follows by combining the last two inequalities.

### A.13 Proof of Theorem 2.7

Consider the following event which implies  $\{Q_1(0) > b\}$  under any scheduling rule: for  $m := \left\lceil \frac{b+1}{L} \right\rceil$  consecutive timeslots before (and including) timeslot 0, there are  $L$  arrivals per timeslots to  $Q_1$ , and all the channels connecting  $Q_1$  to the servers are OFF in each of these timeslots. The probability of this event is  $p^m(1-q)^{nm}$ , and the result follows.

### A.14 Proof of Lemma 2.11

Let  $\ell_t = m$ . By adding packets to the queues if necessary, we ensure  $Q_i(t) = m$  for all  $i$ . By the sample-path-wise dominance property of the iLQF with PullUp algorithm (Lemma 2.8), we know that the probability of overflow of the packet-added system is at least as large as that of the original system.

By the Chernoff bound, in any given timeslot  $t+1$ , we have

$$\mathbb{P} \left( \sum_{i=1}^n A_i^{[n]}(t+1) \geq n\tilde{p}L \right) \leq \exp \{-nH(\tilde{p}|p)\}.$$

Consider the event when in the  $(t+1)^{th}$  timeslot, the number of queues with a nonzero number of arrivals does not exceed  $n\tilde{p}$ . Adding packets to queues if necessary, let the number of queues with  $L$  packet arrivals in the timeslot

$(t + 1)$  be *exactly*  $n\tilde{p}$ . By the sample-path-wise dominance property of the iLQF with PullUp algorithm (Lemma 2.8), by adding packets to the queues we get a system whose overflow probability is larger than the original system. We focus our analysis on this packet-added system.

First round of service:

With probability  $\geq 1 - \exp\{-nH(\tilde{p}|p)\}$ , the queue-length distribution at the beginning of the first round of service is:

Queue-length	Number of queues
$m + L$	$n\tilde{p}$
$m$	$n(1 - \tilde{p})$

There exists a perfect matching between the set of the longest queues and the first  $n\tilde{p}$  servers with probability at least  $1 - 3n\tilde{p}(1 - q)^{n\tilde{p}}$ , by Lemma 2.1. Hence, using this matching for  $\mathcal{M}_*$  in Lemma 2.9, it follows that the first  $n\tilde{p}$  servers are allocated to serve the longest queues in the first round, decreasing their queue-lengths by one each. Thus, the queue-length distribution at the end of the first round of service is:

Queue-length	Number of queues
$m + L - 1$	$n\tilde{p}$
$m$	$n(1 - \tilde{p})$

The servers with indices greater than  $n\tilde{p}$  are available for allocation in the subsequent rounds, if any.

$r^{th}$  round of service, for  $1 < r \leq L$ :

Let the queue-length distribution at the beginning of  $r^{th}$  round of service be:

Queue-length	Number of queues
$m + L - (r - 1)$	$n\tilde{p}$
$m$	$n(1 - \tilde{p})$

Then, by an argument exactly as in the case  $r = 1$ , it follows that with probability at least  $1 - 3n\tilde{p}(1 - q)^{n\tilde{p}}$ , the servers with indices in the set  $\{(r - 1)n\tilde{p} + 1, (r - 1)n\tilde{p} + 2, \dots, rn\tilde{p}\}$  are allocated to serve the longest queues, so that the queue-length distribution at the end of the  $r^{\text{th}}$  round of service is:

Queue-length	Number of queues
$m + L - r$	$n\tilde{p}$
$m$	$n(1 - \tilde{p})$

The servers with indices greater than  $rn\tilde{p}$  are available for allocation in the subsequent rounds, if any.

Further, by Lemma 2.9, the event of finding a bipartite perfect matching between the set of longest queues and the set of servers indexed  $(r - 1)n\tilde{p} + 1$  to  $rn\tilde{p}$  (in the  $r^{\text{th}}$  round) is conditionally independent of all previous rounds, conditioned on the existence of perfect matchings in the earlier rounds, such that for all  $s < r$ , the round  $s$  resulted in allocation of a (perfect) matching between the set of longest queues and the set of servers indexed  $(s - 1)n\tilde{p} + 1$  to  $sn\tilde{p}$ . Hence, considering the first  $L$  rounds of service,

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(t + 1) \leq \max_{1 \leq i \leq n} Q_i(t)\right) \geq (1 - e^{-\Theta_1})(1 - \Theta_2)^L.$$

Therefore,

$$\begin{aligned} \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(t + 1) > \max_{1 \leq i \leq n} Q_i(t)\right) &= 1 - \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(t + 1) \leq \max_{1 \leq i \leq n} Q_i(t)\right) \\ &\leq 1 - (1 - \exp(-\Theta_1))(1 - \Theta_2)^L \\ &\leq 1 - (1 - \exp(-\Theta_1))(1 - L\Theta_2) \end{aligned}$$

$$\begin{aligned}
&= \exp(-\Theta_1) + L\Theta_2 - L\Theta_2 \exp(-\Theta_1) \\
&\leq \exp(-\Theta_1) + L\Theta_2 \\
&= \exp(-nH(\tilde{p}|p)) + L\Theta_2.
\end{aligned}$$

Thus, the proof is complete.

### A.15 Proof of Lemma 2.12

To simplify notation, let  $T = 0$  and  $\ell_0 = m$  in a queuing system  $\underline{Q}$ . Consider a queuing system  $\underline{Q}'$  where  $Q'_i(0) = m$  for all  $i$ , implying  $Q'_i(0) \geq Q_i(0)$  for all  $i$ , and this property continues to hold for all further timeslots if the arrivals and the channel realizations are identical for the two systems (Lemma 2.8). Hereafter in this proof, we will not make references to the system  $\underline{Q}$ .

Fix  $\tilde{p} \in (p, 1/L)$ . The probability that in a given timeslot there are, in all, more than  $n\tilde{p}$  queues with a nonzero number of arrivals is upper bounded by  $\exp(-nH(\tilde{p}|p))$  (by the Chernoff bound). Hence, by union bound, the probability that there are no more than  $n\tilde{p}$  queues with a nonzero number of arrivals in *any* of the  $k$  consecutive timeslots from 1 to  $k$  is at least  $1 - k \exp(-nH(\tilde{p}|p))$ . We condition the rest of the proof on this (high probability) event, and further (if necessary), in every timeslot, we artificially add packets to queues that did not receive packets to enforce the condition that the number of queues receiving  $L$  packets is *exactly*  $n\tilde{p}$ .

Timeslot 1:

After packet arrivals, there are  $n\tilde{p}$  queues each with a length =  $m + L$ , and the rest with a length =  $m$ . Following an analysis similar to that in the proof of Lemma 2.11, there is service for at least  $L$  rounds with probability at least  $(1 - 3n(1 - \tilde{p})(1 - q)^{n(1-\tilde{p})})^L$ , so that after  $L$  rounds of service, all the queues have a length =  $m$  and there exist  $n(1 - \tilde{p}L)$  unallocated servers, with indices  $\{n(1 - \tilde{p}L) + 1, n(1 - \tilde{p}L) + 2, \dots, n\}$ . In the  $(L+1)^{th}$  round of service, there exists a matching of cardinality  $n(1 - \tilde{p}L)$  between the set of unallocated servers and the set of longest queues (i.e. the set of all the queues) with probability at least  $1 - 3n(1 - \tilde{p}L)(1 - q)^{n(1-\tilde{p}L)}$ , so that (with high probability) the queue-length distribution at the end of the first timeslot is:

Queue-length	Number of queues
$m$	$n\tilde{p}L$
$m - 1$	$n(1 - \tilde{p}L)$

Beyond Timeslot 1:

Fix a timeslot  $t_0$ . At the beginning of timeslot  $t_0$ , let the queue-length distribution be as follows:

Queue-length	Number of queues
$m$	$x$
$m - 1$	$n - x$

Fix arbitrary constants  $p' \in (\tilde{p}, 1/L)$  and  $\delta \in (0, (1 - p'L)/2)$ . We prove that if  $\ell_{t_0} = m$ , then the system has served at least  $np'L$  packets in timeslot  $t_0$  with high probability. To this end, consider the queue-length distribution at the beginning of the first round of service, given by:

Queue-length	Number of queues
$m + L$	$y$
$m + L - 1$	$n\tilde{p} - y$
$m$	$x - y$
$m - 1$	$n - x - n\tilde{p} + y$

Define  $z := \max(y, n\delta)$ . In the bipartite graph defined by the set of these longest queues and the set of servers indexed 1 to  $z$ , and the edges defined by the channel realizations, there exists a perfect matching with probability at least

$$1 - 3z(1 - q)^z \stackrel{(a)}{\geq} 1 - 3n\delta(1 - q)^{n\delta},$$

i.e. high probability for  $n$  large. Here, the inequality (a) holds for  $n$  large. Hence, with high probability, the queue-length distribution at the end of the first round of service is:

Queue-length	Number of queues
$m + L - 1$	$n\tilde{p}$
$m$	$x - y$
$m - 1$	$n - x - n\tilde{p} + y$

Further, the servers indexed higher than  $z$  are available for allocations. For the next  $L - 1$  rounds, the system will proceed (with high probability; the analysis being very similar to the first timeslot) to reach the following queue-length distribution:

Queue-length	Number of queues
$m$	$x - y + n\tilde{p}$
$m - 1$	$n - x - n\tilde{p} + y$

At the end of the first  $L$  rounds of service, the total number of packets served equals  $y + (L - 1)n\tilde{p}$  and the servers indexed from  $z + (L - 1)n\tilde{p} + 1$  to  $n$

are available for allocation, and by Lemma 2.9, the existence or non-existence of matchings involving these servers and the set of queues is independent of all the previous  $L$  rounds. Let  $\mathcal{S}^*$  denote the set of the servers indexed from  $z + (L - 1)n\tilde{p} + 1$  to  $n$ .

CASE 1:  $z = y \geq n\delta$ .

$|\mathcal{S}^*| = n - y - (L - 1)n\tilde{p} \geq n - nL\tilde{p} = n(1 - L\tilde{p})$ , since  $y \leq n\tilde{p}$ . Thus, the number of servers available for allocation is at least a constant fraction of  $n$ . Since  $\ell_{t_0} = m$ , and the number of servers available at the beginning of the  $(L + 1)^{th}$  round (where the longest queues are of length  $= m$ ) is at least a constant fraction of  $n$ , we must have the number of longest queues more than the number of available servers, with high probability (using Lemma 2.1). Hence, we have

$$x - y + n\tilde{p} \geq |\mathcal{S}^*| - n\delta \geq nL(p' - \tilde{p}),$$

with high probability. Hence, with high probability (because of Lemma 2.9), the number of packets served in the  $(L + 1)^{th}$  round is at least  $|\mathcal{S}^*| - n\delta \geq nL(p' - \tilde{p})$ . Thus, the total number of packets served in timeslot  $t_0$  is at least  $(n - |\mathcal{S}^*|) + |\mathcal{S}^*| - n\delta = n(1 - \delta) \geq np'L$ , since  $\delta < 1 - p'L$ .

CASE 2:  $z = n\delta \geq y$ .

$|\mathcal{S}^*| = n - n\delta - (L - 1)n\tilde{p} = n(1 - \delta - (L - 1)\tilde{p})$ . Again, since  $\ell_{t_0} = m$ , we have

$$x - y + n\tilde{p} \geq |\mathcal{S}^*| - n\delta \geq nL(p' - \tilde{p}),$$

with high probability. Hence, with high probability (because of Lemma 2.9), the number of packets served in the  $(L + 1)^{th}$  round is at least  $|\mathcal{S}^*| - n\delta \geq$

$nL(p' - \tilde{p})$ . Thus, with high probability, the total number of packets served in timeslot  $t_0$  is at least  $n(L-1)\tilde{p} + |\mathcal{S}^*| - n\delta = n(L-1)\tilde{p} + n(1 - 2\delta - (L-1)\tilde{p}) \geq np'L$ . since  $\delta < (1 - p'L)/2$ .

Further, since  $\ell_{t_0} = m$ , no queues with length  $m - 1$  are served by the specifications of the algorithm, so that at the end of timeslot  $t_0$ , the queue-length distribution remains of the form:

Queue-length	Number of queues
$m$	$x'$
$m - 1$	$n - x'$

Therefore, in timeslot  $t_0$ , if  $\ell_{t_0} = m$ , then the difference between the number of packets served and packet arrivals is at least  $nL(p' - \tilde{p})$  with high probability. Hence, with high probability, for  $k = \lceil \frac{1}{L(p' - \tilde{p})} \rceil$ , there exists a timeslot  $t \leq k$  such that  $\ell_t < m$ , and hence  $\ell_k < m$  with high probability by Lemma 2.4 (and also because the intersection of a finite number of high probability events is a high probability event, which follows from the union bound). For concreteness, choosing  $\tilde{p} = p + \frac{1/L-p}{3}$  and  $p' = p + \frac{2(1/L-p)}{3}$ ,

$$k = \left\lceil \frac{3}{1 - pL} \right\rceil \text{ satisfies } \mathbb{P}(\ell_{T+k} < m \mid \ell_T = m) \geq \frac{1}{2},$$

for all  $n$  large enough, all  $m > 0$  and all  $T$ .

## A.16 Proof of Theorem 2.8

As a result of Lemmas 2.11 and 2.12, and adding dummy packets if necessary, we obtain a system whose queue-length process is a stochastic process with the following properties:

1. In a given timeslot, the maximum queue-length either remains unchanged, or increases by an amount  $L$ . If the timeslot index is a multiple of  $k$  (from Lemma 2.12), then the maximum queue-length may also decrease by 1, in addition to remaining unchanged or increasing by an amount  $L$ .
2. The probability that in a given timeslot, the queue-length increases by an amount  $L$ , is at most

$$2 \max \left( \exp(-nH(\tilde{p}|p)), 3nL\tilde{p}(1-q)^{n\tilde{p}} \right),$$

for all  $\tilde{p} \in (p, 1/L)$ . In particular,  $\tilde{p} = (1/L + p)/2$  is a valid choice.

3. Let  $k$  be the parameter from Lemma 2.12. For an integer  $T$ , if the maximum queue-length in the system at the end of the timeslot  $T$  is positive, then over the next  $k$  timeslots, it decreases by at least 1 with probability at least  $1/2$ .

By the sample-path-wise dominance property of the iLQF with PullUp algorithm (Lemma 2.8), the probability of finite buffer overflow of the modified queue-length process is an upper bound on the probability of finite buffer overflow in the original system. Hence, we analyze the modified process that has the three properties described above.

By an argument similar to the one presented in Appendix A.5, the queue-length process is stable (the Lyapunov function that equals the maximum queue-length works here too). Arguing along the same lines as the

Markov-chain coupling proof in the Appendix A.5, we get for all  $\tilde{p} \in (p, 1/L)$ , in particular for  $\tilde{p} = (1/L + p)/2$ :

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i^{[n]}(0) > b \right) \geq \left\lceil \frac{b+1}{L} \right\rceil \min \left( \tilde{p} \log \frac{1}{1-q}, H(\tilde{p}|p) \right),$$

completing the proof since the right hand side is strictly positive for all  $b \geq 0$ .

## A.17 Proof of Theorem 2.9

The proof is based upon essentially the same ideas and techniques as that of Theorem 2.8, but requires more extensive notation and careful counting. In particular, using Lemma 2.1, we upper-bound the probability that the maximum queue-length increases in a given timeslot, lower-bound the probability that in a (suitable, independent of  $n$ ) constant number of timeslots, the maximum queue-length decreases by at least 1 if it were positive earlier, use sample-path-wise dominance property of the iLQF with PullUp algorithm (Lemma 2.8) to obtain a system where the maximum queue-length change probabilities match the bounds, and use a Markov chain coupling argument to get bounds on the steady-state probabilities of the maximum queue-length. We skip the details.

## Appendix B

### Proofs for Chapter 3

#### B.1 Proof of Lemma 3.1

If  $p\bar{K} > 1$ , then the mean number of packet arrivals to the system in a given timeslot ( $= np\bar{K}$ ) is more than the maximum number of packets that can be served in that timeslot ( $= n$ ), since one server can serve at most one packet. Thus, under any algorithm, the system is unstable.

Consider the case  $p\bar{K} < 1$ . A server  $S_j$  can potentially serve a queue  $Q_i$  in timeslot  $t$  if  $X_{ij}(t) > 0$ . Consider a scheduling rule that, in every timeslot, allocates a server  $S_j$  uniformly at random to one of the queues that it can potentially serve. Elementary calculations (Theorem 2.2) show that the expected number of packets that can be served from any given queue in a given timeslot is  $(1 - (1 - q)^n)$  units. Define  $n_0 = n_0(p, \bar{K}, q) := \left\lceil \frac{\log(1 - p\bar{K})}{\log(1 - q)} \right\rceil$ . Then for all  $n \geq n_0$ , we have  $p\bar{K} < (1 - (1 - q)^n)$ , and every queue (and thus, the system) is stable.

#### B.2 Proof of Lemma 3.2

Fix any  $j \in \{1, 2, \dots, n\}$ . We prove that the contribution of the server  $S_j$  to the weights of the schedules under the iHLQF rule and the MaxWeight

rule cannot differ by more than  $n\hat{\mu}^2$ . Once we prove this, the desired result follows by taking a summation over  $j = 1$  to  $n$ .

1. Suppose that under the iHLQF-rule  $\Lambda$ , the server  $S_j$  is allocated to some queue,  $Q_{b_j}$ . If  $b_j = c_j$ , then we have  $\ell_{b_j} X_{b_j j}(t) = \ell_{c_j} X_{c_j j}(t)$  and the server  $S_j$  contributes the same weight to both the rules, by the definition of  $c_j$ . Hence, we focus on the case  $b_j \neq c_j$ . Further, WLG let  $X_{c_j j}(t) > 0$ .

We first show that under the iHLQF rule, the server  $S_j$  is allocated to serve a queue of length at least  $\ell_{c_j} - n\hat{\mu}$ . Consider the case when  $L = \ell_{c_j}$ , i.e., the round in the algorithm when the queues of length  $\ell_{c_j}$  are considered for server allocation. (If the algorithm never reaches a stage when the queues of length  $= \ell_{c_j}$  are considered for service, then as the condition 1a below makes clear, there is nothing to prove.) The following is an exhaustive list of possibilities regarding  $S_j$  at this stage:

- (a)  $S_j$  is not available for allocation in this round.
- (b)  $S_j$  is allocated in the round when  $L = \ell_{c_j}$ .
- (c)  $S_j$  is available for allocation, but not allocated to serve in this round (when  $L = \ell_{c_j}$ ).

Under the condition 1a, the server  $S_j$  has already been allocated to a queue with length more than  $\ell_{c_j}$  and there is nothing to prove. Under the condition 1b, it serves a queue of length  $\ell_{c_j}$ , and the claim holds. Under condition 1c, the queue  $Q_{c_j}$  must be allocated a server, else the

weight of the matching can be strictly increased by allocating  $S_j$  to  $Q_{c_j}$ . As a result of allocation, the (updated) length of  $Q_{c_j}$  is at least  $\ell_{c_j} - \hat{\mu}$ . Consider the round when  $Q_{c_j}$  is again considered for service. Repeating the above argument, either  $S_j$  is allocated to serve a queue of length  $\geq \ell_{c_j} - \hat{\mu}$ , or  $Q_{c_j}$  is allocated a server, that reduces its length further but no less than  $\ell_{c_j} - 2\hat{\mu}$ . Since there are only  $n$  servers available for allocation, the server  $S_j$  is allocated to serve a queue of length at least  $\ell_{c_j} - n\hat{\mu}$ .

Further, since (by construction) the server  $S_j$  serves a queue  $Q_i$  only if  $X_{ij}(t) \geq X_{c_jj}(t)$ , we have

$$\ell_{b_j} X_{b_jj}(t) \geq (\ell_{c_j} - n\hat{\mu}) X_{c_jj}(t) \geq \ell_{c_j} X_{c_jj}(t) - n\hat{\mu}^2,$$

since  $X_{ij}(t) \leq \hat{\mu}$  for all  $i, j$ .

2. Consider the case when the iHLQF rule  $\Lambda$  does not allocate the server  $S_j$  to any queue. This can only happen if  $\ell_{c_j} \leq (n-1)\hat{\mu}$ , and if all the packets in the queue  $Q_{c_j}$  are served by the allocations under the rule  $\Lambda$ . For if not, then the server  $S_j$  can be allocated to  $Q_{c_j}$ , which is a non-empty queue, contradicting the termination condition of the algorithm. But, the “weight” that the server  $S_j$  contributes to the MaxWeight schedule is  $\ell_{c_j} X_{c_jj}(t)$ , by definition of  $c_j$ . Since  $\ell_{c_j} \leq n\hat{\mu}$  and  $X_{c_jj}(t) \leq \hat{\mu}$ , it follows that the server  $S_j$ ’s contribution to the weight of the MaxWeight schedule is at most  $n\hat{\mu}^2$ .

Thus, we have demonstrated that the contribution of server  $S_j$  to the weights of the schedules under the iHLQF rule and the MaxWeight rule cannot differ by more than  $n\hat{\mu}^2$ , completing the proof.

### B.3 Proof of Lemma 3.3

Step 1 in the definition of the iHLQF-class rules (Definition 3.1) can be implemented in  $\mathcal{O}(n^2)$  computations. Step 2 can be implemented in  $\mathcal{O}(n^2)$  computations. From [9], we know that in an edge-weighted bipartite graph (in fact, in any edge-weighted graph with  $\mathcal{O}(n)$  nodes), the maximum weight matching can be found in  $\mathcal{O}(n^3)$  computations, implying that step 3 can be implemented in  $\mathcal{O}(n^3)$  computations (since tie-breaking between the maximum weight matchings can be arbitrary). In step 4, instead of decrementing  $L$  by 1, find the length  $L'$  of the longest queue(s) whose length is at most  $L - 1$ , and set  $L = L'$ . This modification results in the same allocation decisions as the original algorithm, resulting in Step 4 requiring  $\mathcal{O}(n)$  computations. Further, every execution of steps 3 and 4 removes at least one server or one queue from further consideration (by allocating a server, or by not being able to allocate any server to a queue of length  $L$ , leaving its length at  $L > L'$ ). To begin with, the system has  $2n$  servers + queues, implying that the steps 3 and 4 together take  $\mathcal{O}(n^4)$  operation overall. Finally the output can be reported in  $\mathcal{O}(n)$  computations (memory reads) by keeping track of the allocations in a server allocation vector. Thus, the algorithm can be implemented in  $\mathcal{O}(n^4)$  computations per timeslot.

## B.4 Proof of Theorem 3.3

Fix any integer  $T$ , and consider the queues at the end of timeslot  $T$ . Define  $\gamma_n := 1 - e^{-\sqrt{n}}$ , and  $p' := p/2$ .

### Timeslot $T + 1$ :

By the Chernoff bound, there exists an integer  $n_1$  such that for all  $n \geq n_1$ , with probability at least<sup>1</sup>  $\gamma_n$ , at least  $np'$  queues see arrivals in timeslot  $T + 1$ . Define  $\alpha := -2/\log(1 - q)$ . Fix an integer  $N$  such that for all  $n \geq N$ , we have  $\alpha \log n \geq 1$ . Define  $\beta := \alpha \log n$ , and consider the first  $\beta$  queues in the order of priority for service (after arrivals). In particular, the first queue in the order of priority is the longest queue with the smallest index, the second one is the longest queue with the second smallest index or the second longest queue with the smallest index, and so on. Let the set of these queues be  $\mathcal{Q}^* := \{Q_{i_1}, Q_{i_2}, \dots, Q_{i_\beta}\}$ . Let  $E_j$  denote the event that server  $S_j$  is not connected with any of the queues in  $\mathcal{Q}^*$ . Then, since  $\mathbb{P}(E_j) = (1 - q)^\beta = (1 - q)^{\alpha \log n}$ , we have

$$\mathbb{P}\left(\bigcup_{j=1}^n E_j\right) \leq \sum_{j=1}^n \mathbb{P}(E_j) = n(1 - q)^{\alpha \log n} = \frac{1}{n}.$$

Thus, with probability at least  $1 - 1/n$ , each one of the servers is connected to a queue in  $\mathcal{Q}^*$ . By the definition of the MaxWeight rule, a server connected to one of the queues in  $\mathcal{Q}^*$  is allocated to one of the queues in  $\mathcal{Q}^*$ . Since  $|\mathcal{Q}^*| = \beta$  and at least  $np'$  queues had packet arrivals, it follows that at

---

<sup>1</sup>The Chernoff bound actually gives a much stronger result, with the probability of the desired event being at least  $1 - e^{-cn}$  for some constant  $c > 0$ . But a weaker result implied by the Chernoff bound, as stated here, suffices for our purpose.

the end of timeslot  $T + 1$ , the system has at least  $np' - \alpha \log n$  queues at length 1. By the union bound (for  $n \geq \max(N, n_1)$ ), the probability of this event is at least  $1 - (1/n + e^{-\sqrt{n}})$ . Let this set of queues (of length at least 1) be called  $\mathcal{A}_1$ .

**Timeslot  $T + 2$ :**

The arrivals in the timeslot  $T + 2$  are independent of all the random variables involved in the definition of the set  $\mathcal{A}_1$ . Thus, by appropriately using the Chernoff bound, there exists an integer  $n_2$  such that for all  $n \geq n_2$ , with probability at least  $\gamma_n$ , at least  $p'$  fraction of queues in the set  $\mathcal{A}_1$  see arrivals in timeslot  $T + 2$ . By an argument similar to that for Timeslot  $T + 1$ , it follows that, with probability at least  $1 - 1/n$ , no more than  $\alpha \log n$  of the queues receive service. Combining the results for the timeslots and using the union bound, we have the following conclusion: for all  $n \geq \max(N, n_1, n_2)$ , with probability at least  $1 - 2(1/n + e^{-\sqrt{n}})$ , there exists a set  $\mathcal{A}_2$  of queues such that:

- $|\mathcal{A}_2| \geq p'(np' - \alpha \log n) - \alpha \log n \geq np'^2 - 2\alpha \log n$ .
- Each queue in  $\mathcal{A}_2$  has a length at least 2.

Continuing this way (formally, by induction), the following claim holds: at the end of timeslot  $T + b + 1$ , for  $n \geq \max(N, n_1, n_2, \dots, n_{b+1})$ , with probability at least  $1 - (b + 1)(1/n + e^{-\sqrt{n}})$ , there exists a set  $\mathcal{A}_{b+1}$  of queues such that:

- $|\mathcal{A}_{b+1}| \geq p'(np^b - b\alpha \log n) - \alpha \log n$ , implying  $|\mathcal{A}_{b+1}| \geq np^{b+1} - (b + 1)\alpha \log n$ .
- Each queue in  $\mathcal{A}_{b+1}$  has a length at least  $b + 1$ .

There exists an integer  $n_0$  such that for all  $n \geq n_0$ , we have  $np^{b+1} - (b + 1)\alpha \log n \geq 1$  and  $(b + 1)(1/n + e^{-\sqrt{n}}) \leq 1/2$ . Hence, for a system with  $n \geq \max(N, n_0, n_1, \dots, n_{b+1})$ , with at least a probability  $1/2$  and starting with any initial configuration of queue-lengths, we have a queue of length  $b + 1$  at the end of a further  $b + 1$  timeslots. Thus even for large  $n$ , the small buffer overflow event occurs with at least a constant probability, and the proof is complete.

## B.5 Proof of Lemma 3.4

Let Assumption 3.2 hold. In a given timeslot  $t$ , the MaxWeight rule needs to find, for every server  $S_j$ , a queue  $Q_i$  that maximizes the product of the instantaneous channel rate  $X_{ij}(t)$  and the length of queue  $Q_i$  after packet arrivals (if any). Thus, for every server, the rule needs to perform  $n$  multiplications. All the  $n$  multiplications are necessary: not performing even one of these multiplications (in the worst case) can lead to an incorrect allocation. Further, since all the channels are mutually independent, each server needs  $n$  separate computations, i.e., the calculations not involving  $X_{.j}(t)$  are useless for  $S_j$ . Since there are  $n$  servers in the system, any implementation of the MaxWeight rule requires  $\Omega(n^2)$  computations per timeslot.

## B.6 Proof of Lemma 3.5

Fix any  $j \in \{1, 2, \dots, n\}$ . We prove that the contribution of the server  $S_j$  to the weights of the schedules under the SSG rule and the MaxWeight rule cannot differ by more than  $n\hat{\mu}^2$ . Once we prove this, the desired result follows by taking a summation over  $j = 1$  to  $n$ .

1. Suppose that under the SSG rule, the server  $S_j$  is allocated to some queue  $Q_{a_j}$ , while under the MaxWeight rule, it is allocated to  $Q_{b_j}$ . If  $a_j = b_j$ , then we have  $\ell_{a_j} X_{a_j j}(t) = \ell_{b_j} X_{b_j j}(t)$  and the server  $S_j$  contributes the same weight to both the rules. Hence, we focus on the case  $a_j \neq b_j$ . In this case,  $Q_{b_j}$  is connected to server  $S_j$ , but does not get served by  $S_j$  under the SSG rule. It follows that

$$\begin{aligned} X_{a_j j}(t) \ell_{a_j} &\stackrel{(a)}{\geq} X_{a_j j}(t) Q_{a_j}^{(j-1)}(t) \\ &\stackrel{(b)}{\geq} X_{b_j j}(t) Q_{b_j}^{(j-1)}(t) \\ &\stackrel{(c)}{\geq} X_{b_j j}(t) (\ell_{b_j} - n\hat{\mu}) \\ &\geq X_{b_j j}(t) \ell_{b_j} - n\hat{\mu}^2. \end{aligned}$$

Here, the inequality (a) holds because  $\ell_{a_j} = Q_{a_j}^{(0)}(t) \geq Q_{a_j}^{(j-1)}(t)$ , since queue-lengths can only monotonically decrease as the successive rounds proceed (recall that the arrivals occur before round 1). The inequality (b) holds because in round  $j$ , the SSG rule allocates server  $S_j$  to a queue that maximizes the product of the channel-rate and queue-length. The inequality (c) holds because any given queue can receive at most  $\hat{\mu}$  units

of service in a given round (of SSG), implying that the length of  $Q_{b_j}$  after  $j - 1$  rounds is at least  $\ell_{b_j} - (j - 1)\hat{\mu} \geq \ell_{b_j} - n\hat{\mu}$ . It follows that

$$X_{a_j j}(t)\ell_{a_j} \geq X_{b_j j}(t)\ell_{b_j} - n\hat{\mu}^2.$$

2. Consider the case when the SSG rule does not allocate the server  $S_j$  to any queue, but the MaxWeight rule assigns it to the queue  $Q_{b_j}$ . This can only happen if  $\ell_{b_j} \leq (j - 1)\hat{\mu}$ , and if all the packets in the queue  $Q_{a_j}$  are served by the allocations under the SSG rule, before the server  $S_j$  is considered for service in the  $j^{\text{th}}$  round. For if not, then the server  $S_j$  can be allocated to  $Q_{b_j}$ , which is a non-empty queue, contradicting the definition of the algorithm. But, the “weight” that the server  $S_j$  contributes to the MaxWeight schedule is  $\ell_{b_j} X_{b_j j}(t)$ . Since  $\ell_{b_j} \leq n\hat{\mu}$  and  $X_{b_j j}(t) \leq \hat{\mu}$ , it follows that the server  $S_j$ ’s contribution to the weight of the MaxWeight schedule is at most  $n\hat{\mu}^2$ .

Thus, we have demonstrated that the contribution of server  $S_j$  to the weights of the schedules under the SSG rule and the MaxWeight rule cannot differ by more than  $n\hat{\mu}^2$ , completing the proof.

## B.7 Proof of Theorem 3.5

The proof of this Theorem is on the same lines as that of Theorem 1 in [10], which establishes the following: consider a service rule  $R$  that, in every timeslot, picks a schedule whose weight is an arbitrarily high fraction of that

of the MaxWeight schedule whenever the sum of the queue lengths is large enough. Then, the rule  $R$  makes the system stable in the mean, and positive recurrent under the stated conditions. That proof applies almost unchanged when the weight of the schedule selected by the candidate policy  $R$  is at most an additive constant smaller than the maximum possible, provided that this constant is independent of the queue-lengths and is a function of system parameters only. This result and Lemma 3.5 complete the proof.

## B.8 Proof of Lemma 3.6

We need to prove that  $Q_i^{(n)}(t+1) \leq R_i^{(n)}(t+1)$  for all  $i$ . Suppose that for some  $k \in \{1, 2, \dots, n\}$ , we have

$$Q_i^{(k-1)}(t+1) \leq R_i^{(k-1)}(t+1), \quad \forall i.$$

We need to prove that  $Q_i^{(k)}(t+1) \leq R_i^{(k)}(t+1)$  for all  $i$ , and the proof would be complete by the principle of mathematical induction.

Let in the system  $\underline{R}$ , server  $S_k$  be allocated to serve queue  $R_j$ . If (in the system  $\underline{Q}$ ) the server  $S_k$  is allocated to serve  $Q_j$ , then there is nothing to prove. If  $S_k$  is not allocated to serve  $Q_j$ , it must be because of one of the following reasons:

1.  $Q_j^{(k-1)}(t+1) < Q_r^{(k-1)}(t+1)$  for some  $r$ , and  $S_k$  is connected to  $Q_r$ , i.e.,  $X_{rk}(t+1) = 1$ .

2.  $Q_j^{(k-1)}(t+1) = Q_m^{(k-1)}(t+1)$  for some  $m < j$ , and the queues  $Q_j, Q_m$  are both among the longest queues connected to  $S_k$ , so according to the tie-breaking rule, queue  $Q_m$  is served in the  $k^{\text{th}}$  round.

In case 1, we have (by hypothesis)

$$Q_j^{(k-1)}(t+1) < Q_r^{(k-1)}(t+1) \leq R_r^{(k-1)}(t+1),$$

and  $R_r^{(k-1)}(t+1) \leq R_j^{(k-1)}(t+1)$  by the definition of the SSG rule. (Otherwise,  $S_k$  would have been allocated to serve  $R_r$  because  $X_{rk}(t+1) = 1$ .) Hence, irrespective of the allocation of  $S_k$  (in the system  $\underline{Q}$ ), we have  $Q_j^{(k)}(t+1) \leq R_j^{(k)}(t+1)$ , and consequently  $R_i^{(k)}(t+1) \geq Q_i^{(k)}(t+1)$  for all  $i$ .

In case 2, we must have

$$Q_j^{(k-1)}(t+1) < R_j^{(k-1)}(t+1),$$

because if  $Q_j^{(k-1)}(t+1) = R_j^{(k-1)}(t+1)$ , then

$$R_j^{(k-1)}(t+1) = Q_j^{(k-1)}(t+1) = Q_m^{(k-1)}(t+1) \leq R_m^{(k-1)}(t+1),$$

and  $m < j$  implies that in the system  $\underline{R}$ , server  $S_k$  must have been allocated to  $R_m$  and not  $R_j$ . Therefore, we have  $Q_j^{(k-1)}(t+1) < R_j^{(k-1)}(t+1)$ , and hence  $Q_j^{(k)}(t+1) \leq R_j^{(k)}(t+1)$ , implying  $R_i^{(k)}(t+1) \geq Q_i^{(k)}(t+1)$  for all  $i$ .

## B.9 Proof of Lemma 3.8

Define  $L(t) := \max_{1 \leq i \leq n} Q_i(t)$ . For some timeslot  $t$ , let  $L(t) = m$ . Adding dummy packets if necessary, we consider a system where all the queues

are of length  $m$  at the end of timeslot  $t$ , and by doing so, we get only a “worse” system, a system with sample-pathwise longer queues for all future times, thanks to Lemma 3.6.

Now, fix any  $p' \in (p, 1)$ . By the Chernoff bound, there exists an integer  $n_0$  such that for all  $n \geq n_0$ , the probability that in timeslot  $t + 1$ , more than  $np'$  queues have arrivals is no more than  $\exp(-nH(p'|p))$ . We condition on the (high probability) event of having at most  $np'$  queues with a nonzero number of arrivals. Again, adding packets if necessary, ensure that *exactly*  $np'$  queues see arrivals in timeslot  $t + 1$ , so that after arrivals, the queue-length profile is the following:

Queue-length	Number of queues
$m + 1$	$np'$
$m$	$n(1 - p')$

Fix any  $\delta \in (0, \frac{q(1-p')}{2-q})$ , so that  $2\delta/q < 1 - p' + \delta$ . Let  $Z$  denote the event that the first  $n(p' - \delta)$  servers each serve a longest queue (i.e., a queue of length  $m + 1$ ). Then,

$$\begin{aligned}
\mathbb{P}(Z^c) &\leq \sum_{i=1}^{n(p'-\delta)} \mathbb{P}(S_i \text{ does not serve a longest queue} \mid S_j \\
&\quad \text{served a longest queue, } \forall j < i) \\
&= \sum_{i=1}^{n(p'-\delta)} \mathbb{P}(\text{All the channels connecting } S_i \text{ to the} \\
&\quad \text{remaining longest queues are OFF}) \\
&= \sum_{i=1}^{n(p'-\delta)} (1 - q)^{np' - (i-1)} \leq n(1 - q)^{n\delta}.
\end{aligned}$$

Therefore, for  $n$  large enough,  $Z$  is a high probability event. Conditioned on  $Z$ , the queue-length profile at the end of exactly  $n(p' - \delta)$  rounds of service is the following:

Queue-length	Number of queues
$m + 1$	$n\delta$
$m$	$n(1 - \delta)$

Consider the system configuration after  $n(p' - \delta)$  rounds of service. There are  $n(1 - p' + \delta)$  servers left for allocation, which (because of the choice of  $\delta$ ) is more than  $2n\delta/q$ . By the Chernoff bound and after a little algebra, the probability that a given longest queue is connected to less than  $n\delta$  of the remaining servers is at most  $\exp(-2n\delta H(\frac{q}{2}|q)/q)$ , implying that (by the union bound) the probability that any one of the longest queues is connected to less than  $n\delta$  servers is at most  $n\delta \exp(-2n\delta H(\frac{q}{2}|q)/q)$ . Thus, by Lemma 3.7, with probability at least  $1 - n\delta \exp(-2n\delta H(\frac{q}{2}|q)/q)$ , all the remaining longest queues are served.

Therefore, by the union bound, the probability that in a given timeslot, at least one of the longest queues is not served is no more than  $\exp(-nH(p'|p)) + n(1 - q)^{n\delta} + n\delta \exp(-2n\delta H(\frac{q}{2}|q)/q)$ , completing the proof.

## B.10 Proof of Lemma 3.9

We first prove an auxiliary claim.

**Claim B.1.** *Let the queue-length profile at the beginning of timeslot  $T$  be the following:*

Queue-length	Number of queues
$m$	$\alpha$
$m - 1$	$n - \alpha$

Fix  $p' \in (p, 1)$  and  $\delta \in (0, \frac{q(1-p')}{2(2-q)})$ . In particular, let  $p' = (1 + p)/2$  and  $\delta = \frac{q(1-p')}{5}$ . Define  $\epsilon := q(1 - p' - 2\delta(\frac{2}{q} - 1))/2$ . Then, there exists  $\theta > 0$  and an integer  $n_0$  such that for all  $n \geq n_0$ , with probability at least  $1 - e^{-n\theta}$  (and adding dummy packets if necessary), the queue-length profile at the beginning of the timeslot  $T + 1$  is

Queue-length	Number of queues
$m$	$(\alpha - n\epsilon)^+$
$m - 1$	$n - (\alpha - n\epsilon)^+$

The proof of this claim is based upon an intersection of a finite number of high probability events (each of which occurs with probability at least  $1 - e^{-n\gamma}$  for some  $\gamma$ ), which is itself a high probability event.

*Proof of Claim B.1.* Fix  $p' \in (p, 1)$ . By the Chernoff bound, there exists an integer  $n_1$  such that for all  $n \geq n_1$ , with probability at least  $1 - \exp(-nH(p'|p))$ , the number of queues that see arrivals in timeslot  $T$  is no more than  $np'$ . Adding dummy packets if necessary, we ensure that the number of queues seeing arrivals is *exactly*  $np'$ . Let  $x$  of the queues already at length  $m$  have packet arrivals. Thus, the queue-length profile after arrivals is

Queue-length	Number of queues
$m + 1$	$x$
$m$	$\alpha - x + np' - x$
$m - 1$	$n - \alpha - (np' - x)$

Following an argument similar to that in the proof of Lemma 3.8, it follows that there exists  $\theta_1 > 0$  and an integer  $n_2$  such that if  $n \geq n_2$ , then with probability at least  $1 - e^{-n\theta_1}$ , at the end of  $x + n\delta(\frac{2}{q} - 1)$  rounds of service, the queue-length profile is the following (or, can be obtained by adding packets):

Queue-length	Number of queues
$m$	$\alpha + np' - x$
$m - 1$	$n - \alpha - (np' - x)$

Thus, again by an argument similar to that in the proof of Lemma 3.8, there exists  $\theta_2 > 0$  and an integer  $n_3$  such that if  $n \geq n_3$ , then with probability at least  $1 - e^{-n\theta_2}$ , after a further  $np' - x + n\delta(\frac{2}{q} - 1)$  rounds of service, the queue-length profile is the following (or, can be obtained by adding dummy packets):

Queue-length	Number of queues
$m$	$\alpha$
$m - 1$	$n - \alpha$

At this stage, we have  $n(1 - p' - 2\delta(\frac{2}{q} - 1))$  servers left for allocation. Therefore, by an argument similar to that in the proof of Lemma 3.8, there exists  $\theta_3 > 0$  and an integer  $n_4$  such that if  $n \geq n_4$ , then with probability at least  $1 - e^{-n\theta_3}$ , at least  $nq(1 - p' - 2\delta(\frac{2}{q} - 1))/2 = n\epsilon$  of the longest queues are served (unless  $\alpha < n\epsilon$ , in which case all the  $\alpha$  of the longest queues are served). Thus, by the union bound, the probability that we have (or can obtain by adding dummy packets) the desired queue-length profile is at least  $1 - (e^{-nH(p'|p)} + e^{-n\theta_1} + e^{-n\theta_2} + e^{-n\theta_3})$ . Define  $\theta = \frac{1}{2} \min(\frac{H(p'|p)}{2}, \theta_1, \theta_2, \theta_3)$ . Then, there exists an integer  $n_5$  such that

for all  $n \geq n_5$ , we have

$$1 - (e^{-nH(p'|p)} + e^{-n\theta_1} + e^{-n\theta_2} + e^{-n\theta_3}) \geq 1 - e^{-n\theta}.$$

Thus,  $n_0 = \max(n_1, n_2, n_3, n_4, n_5)$  is a valid choice for  $n_0$ , completing the proof.  $\square$

(Proof of Lemma 3.9)

WLG let  $t = 0$ , and let  $L(0) = m$ . By Lemma 3.6, we can add dummy packets to any of the queues and get a dominant system for all future timeslots. Hence, adding dummy packets if necessary, we consider a system where at the beginning of the first timeslot, all the queues have a length  $= m$ . We make no further references to the original queue-lengths and refer the queues and the queue-lengths in the “new” system by  $Q_i$  and  $Q_i(t)$  respectively. Thus, at the beginning of the first timeslot, the queue-length profile is as desired for Claim B.1 to hold, with  $\alpha = n$ . Therefore, at the end of at most  $k = \lceil \frac{1}{\epsilon} \rceil$  timeslots and (by the union bound) with probability at least  $1 - ke^{-n\theta}$ , the maximum queue-length in the system is no more than  $m - 1$ . Since  $k$  is independent of  $n$  and  $\theta > 0$ , we can choose  $n$  large enough so that  $1 - ke^{-n\theta} > 1/2$ . This completes the proof.

## B.11 Proof of Theorem 3.8

Consider the following implementation of the SSG rule: the system maintains a vector of queue-lengths  $[Q_1, Q_2, \dots, Q_n]$  and updates it as arrivals

and service occur. It also maintains a vector that records the queue-index that server  $S_k$  is allocated to, and updates it in every round. Updating the queue-length vector after arrivals (Step 1 in the definition of SSG) involves  $n$  additions and  $n^2$  initializations and can be implemented in  $\mathcal{O}(n^2)$  computations. In Step 2, any given round can be implemented in  $\mathcal{O}(n)$  computations, and therefore the entire Step 2 ( $n$  rounds) can be implemented in  $\mathcal{O}(n^2)$  computations. Step 3 can be implemented with  $\mathcal{O}(1)$  computations. Finally the output can be reported in  $\mathcal{O}(n)$  computations (memory reads), by keeping track of the allocations in a server allocation vector. Hence, the overall complexity is  $\mathcal{O}(n^2)$  computations per timeslot.

# Appendix C

## Proofs for Chapter 4

### C.1 Proof of Lemma 4.1

If  $\lambda := \sum_{m=1}^M mp_m > K$ , then the mean number of packet arrivals to the system in a given timeslot ( $= n\lambda$ ) is more than the maximum number of packets that can be served in that timeslot ( $= Kn$ ), since one server can serve at most  $K$  packets. Thus, under any algorithm, the system is unstable.

Consider the case  $\lambda < K$ . A server  $S_j$  can potentially serve a queue  $Q_i$  in timeslot  $t$  if  $X_{ij}(t) > 0$ . Consider a scheduling rule that, in every timeslot, allocates a server  $S_j$  uniformly at random to one of the queues that it can potentially serve with a rate  $= K$ . Elementary calculations (Theorem 2.2) show that the expected number of packets that can be served from any given queue in a given timeslot is  $K(1 - (1 - q)^n)$  units. Define  $n_0 = n_0(p, M, K, q) := \left\lceil \frac{\log(1 - \lambda/K)}{\log(1 - q)} \right\rceil$ . Then, for all  $n \geq n_0$ , we have  $\lambda < K(1 - (1 - q)^n)$ , and every queue (and thus, the system) is stable.

### C.2 Proof of Theorem 4.1

Consider the following event which implies  $\{Q_1(0) > b\}$  under any scheduling rule: for  $m := \left( \left\lfloor \frac{b}{M} \right\rfloor + 1 \right)$  consecutive timeslots before (and in-

cluding) timeslot 0, there are  $M$  arrivals per timeslot to  $Q_1$ , and all the channels connecting  $Q_1$  to the servers are OFF in each of these timeslots. The probability of this event is  $p_M^n (q_0)^{nm}$ , and the result follows.

### C.3 Proof of Lemma 4.2

If  $pM > 1$ , then the mean number of packet arrivals to the system in a given timeslot ( $= npM$ ) is more than the maximum number of packets that can be served in that timeslot ( $= n$ ), since one server can serve at most 1 packet in a given timeslot. Thus, under any algorithm, the system is unstable.

Consider the case  $pM < 1$ . Let  $\gamma := \min(\alpha, 1 - \beta)$ . Then regardless of the value of  $X_{ij}(t - 1)$ , we have  $X_{ij}(t) = 1$  with probability at least  $\gamma$ . A server  $S_j$  can potentially serve a queue  $Q_i$  in timeslot  $t$  if  $X_{ij}(t) > 0$ . Consider a scheduling rule that, in every timeslot, allocates a server  $S_j$  uniformly at random to one of the queues that it can potentially serve. Elementary calculations (Theorem 2.2) show that the expected number of packets that can be served from any given queue in a given timeslot is at least  $(1 - (1 - \gamma)^n)$  units. Define  $n_0 = n_0(p, M, \alpha, \beta) := \left\lceil \frac{\log(1 - pM)}{\log(1 - \min(\alpha, 1 - \beta))} \right\rceil$ . Then, for all  $n \geq n_0$ , we have  $pM < (1 - (1 - \gamma)^n)$ , and every queue (and thus, the system) is stable.

## C.4 Proof of Theorem 4.2

Consider the following event which implies  $\{Q_1(0) > b\}$  under any scheduling rule: for  $m := \left(\left\lfloor \frac{b}{M} \right\rfloor + 1\right)$  consecutive timeslots before (and including) timeslot 0, there are  $M$  arrivals per timeslot to  $Q_1$ , and all the channels connecting  $Q_1$  to the servers are OFF in each of these timeslots. The probability of  $M$  arrivals per timeslot for  $m$  consecutive timeslots is  $p^m$ . Regardless of the channel realizations in the previous timeslot(s), any given channel is OFF (i.e., in the state 0) in the current timeslot  $t$  with probability at least  $\min(\beta, 1 - \alpha)$ . Once the channel is in the OFF state, the probability of it remaining OFF for the next  $m - 1$  timeslots is  $(1 - \alpha)^{m-1}$ . Since there are  $n$  i.i.d. channels connected to  $Q_1$ , the probability that each one of them is OFF in the  $m$  timeslots leading to and including the timeslot 0 is at least  $(\min(\beta, 1 - \alpha) \cdot (1 - \alpha)^{m-1})^n$ , and the result follows.

## C.5 Proof of Lemma 4.3

If possible, let

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > 0 \right) = \delta > 0$$

for some  $\delta > 0$  under some scheduling algorithm. Then for  $n$  large enough, we have

$$\mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > 0 \right) \leq e^{-n\delta/2},$$

implying that the long-term average number of timeslots for which we have  $\max_{1 \leq i \leq n} Q_i(t) = 0$  is at least  $1 - e^{-n\delta/2}$ , or greater than 0.8 for  $n$  large enough.

Suppose at the end of some timeslot  $T$ , we have  $\max_{1 \leq i \leq n} Q_i(T) = 0$ .

Fix  $\epsilon > 0$  such that

$$[p'_0, p'_1, \dots, p'_M] = [p_0 + M\epsilon, p_1 - \epsilon, p_2 - \epsilon, \dots, p_M - \epsilon]$$

is a probability vector with strictly positive components, and

$$\sum_{m=1}^M p'_m \left\lceil \frac{m}{K} \right\rceil > 1.$$

Let  $n_m$  denote the number of queues that receive  $m$  packets in timeslot  $T + 1$ . By Sanov's theorem ([7], Theorem 2.1.10), we know that there exists  $\zeta > 0$  such that for  $n$  large enough,

$$\mathbb{P} \left( \frac{n_m}{n} \geq p'_m \quad \forall m \in \{1, 2, \dots, M\} \right) \geq 1 - e^{-n\zeta}.$$

For a queue with length  $= m$ , any algorithm must allocate at least  $\left\lceil \frac{m}{K} \right\rceil$  servers to serve all packets from it. But if the algorithm tries to allocate this way, then (even assuming that each of the servers has a channel with rate  $= K$  to serve any one of the queues), the number of servers necessary to drain all the packets is

$$\sum_{m=1}^M n_m \left\lceil \frac{m}{K} \right\rceil \geq n \sum_{m=1}^M p'_m \left\lceil \frac{m}{K} \right\rceil > n.$$

Hence, if at the end of timeslot  $T$  we have  $\max_{1 \leq i \leq n} Q_i(T) = 0$ , then at the end of timeslot  $T + 1$ , with probability at least  $1 - e^{-n\zeta}$ , we have  $\max_{1 \leq i \leq n} Q_i(T + 1) \geq 1$  for some  $\zeta > 0$ . It follows that the long-term fraction of timeslots for which  $\max_{1 \leq i \leq n} Q_i(t) = 0$  is no more than  $\frac{1}{2 - (1 - e^{-n\zeta})} = \frac{1}{1 + e^{-n\zeta}} < \frac{2}{3}$ , for  $n$  large enough. This contradicts the earlier claim that the long-term fraction of timeslots for which  $\max_{1 \leq i \leq n} Q_i(t) = 0$  is at least 0.8, completing the proof.

## C.6 Proof of Theorem 4.3

A stationary distribution of  $X^*(t)$  is given by

$$\mathbb{P}(X^*(t) = x) = \mathbb{P}(\mathbf{X}(t) \in \mathcal{W}_x) = \sum_{z \in \mathcal{W}_x} \sigma(z).$$

We show that this is a stationary distribution of the Markov chain  $Y(t)$ . By the detailed flow-balance for the Markov chain  $\mathbf{X}(t)$ , for every non-negative integer  $x$ , we have

$$\sum_{\substack{z \in \mathcal{W}_x \\ y \geq 0 \\ y \neq x}} \sum_{v \in \mathcal{W}_y} \sigma(z) p(z, v) = \sum_{\substack{z \in \mathcal{W}_x \\ y \geq 0 \\ y \neq x}} \sum_{v \in \mathcal{W}_y} \sigma(v) p(v, z). \quad (\text{C.6.1})$$

In order that  $\mathbb{P}(Y(t) = x) = \mathbb{P}(X^*(t) = x) = \sum_{z \in \mathcal{W}_x} \sigma(z)$ , it is necessary and sufficient that the proposed stationary distribution satisfies the detailed flow-balance equations for the Markov chain  $Y(t)$ , is non-negative and adds to 1. In other words, for every non-negative integer  $x$ , we want

$$\begin{aligned} & \sum_{\substack{y \geq 0 \\ y \neq x}} \left( \sum_{z \in \mathcal{W}_x} \sigma(z) \right) \mathbb{P}(Y(t+1) = y \mid Y(t) = x) \\ &= \sum_{\substack{y \geq 0 \\ y \neq x}} \left( \sum_{z \in \mathcal{W}_y} \sigma(z) \right) \mathbb{P}(Y(t+1) = x \mid Y(t) = y). \end{aligned}$$

We now verify that the proposed stationary distribution  $\mathbb{P}(Y(t) = x) =$

$\mathbb{P}(X^*(t) = x) = \sum_{z \in \mathcal{W}_x} \sigma(z)$  satisfies this condition, as follows:

$$\begin{aligned}
& \sum_{\substack{y \geq 0 \\ y \neq x}} \left( \sum_{z \in \mathcal{W}_x} \sigma(z) \right) \mathbb{P}(Y(t+1) = y \mid Y(t) = x) \\
&= \mathbb{P}(\mathbf{X}(t) \in \mathcal{W}_x) \sum_{\substack{y \geq 0 \\ y \neq x}} \mathbb{P}(\mathbf{X}(t+1) \in \mathcal{W}_y \mid \mathbf{X}(t) \in \mathcal{W}_x) \\
&= \sum_{\substack{y \geq 0 \\ y \neq x}} \mathbb{P}(\mathbf{X}(t+1) \in \mathcal{W}_y, \mathbf{X}(t) \in \mathcal{W}_x) \\
&= \sum_{\substack{y \geq 0 \\ y \neq x}} \sum_{z \in \mathcal{W}_x} \sum_{v \in \mathcal{W}_y} \sigma(z) p(z, v) \\
&\stackrel{(a)}{=} \sum_{\substack{y \geq 0 \\ y \neq x}} \sum_{z \in \mathcal{W}_x} \sum_{v \in \mathcal{W}_y} \sigma(v) p(v, z) \\
&= \sum_{\substack{y \geq 0 \\ y \neq x}} \mathbb{P}(\mathbf{X}(t+1) \in \mathcal{W}_x, \mathbf{X}(t) \in \mathcal{W}_y) \\
&= \sum_{\substack{y \geq 0 \\ y \neq x}} \mathbb{P}(\mathbf{X}(t) = y) \mathbb{P}(\mathbf{X}(t+1) = x \mid \mathbf{X}(t) = y) \\
&= \sum_{\substack{y \geq 0 \\ y \neq x}} \left( \sum_{z \in \mathcal{W}_y} \sigma(z) \right) \mathbb{P}(Y(t+1) = x \mid Y(t) = y),
\end{aligned}$$

where the step (a) follows from Equation (C.6.1). Note also that  $\sum_{z \in \mathcal{W}_x} \sigma_z \geq 0$  and  $\sum_{x \geq 0} \sum_{z \in \mathcal{W}_x} \sigma_z = 1$ , since the sets  $\mathcal{W}_x$  form a partition of the state-space of the original Markov chain  $\mathbf{X}(t)$ . Hence, the proof is complete.

## C.7 Proof of Theorem 4.4

The proof proceeds by induction. Let the claim hold for some  $t = k - 1 \geq 0$ , i.e.,  $Y(k - 1) \leq_{st} W(k - 1)$ . We have

$$\begin{aligned} Y(k) &= Y(k - 1) + Z_{k-1}(Y(k - 1)), \\ W(k) &= W(k - 1) + \tilde{Z}_{k-1}(W(k - 1)). \end{aligned}$$

By a standard result in stochastic ordering, there exist random variables  $\hat{Y}(k - 1)$  and  $\hat{W}(k - 1)$  such that

$$\hat{Y}(k - 1) \stackrel{d}{=} Y(k - 1), \quad \hat{W}(k - 1) \stackrel{d}{=} W(k - 1),$$

and 
$$\hat{Y}(k - 1) \leq \hat{W}(k - 1), \quad a.s.$$

Define 
$$\begin{aligned} \hat{Y}(k) &:= \hat{Y}(k - 1) + Z_{k-1}(\hat{Y}(k - 1)), \\ \hat{W}(k) &:= \hat{W}(k - 1) + \tilde{Z}_{k-1}(\hat{W}(k - 1)). \end{aligned}$$

Now,

$$\mathbb{P}(W(k) = j) = \sum_i \mathbb{P}(i + \tilde{Z}_{k-1}(i) = j) \mathbb{P}(W(k - 1) = i),$$

and

$$\mathbb{P}(\hat{W}(k) = j) = \sum_i \mathbb{P}(i + \tilde{Z}_{k-1}(i) = j) \mathbb{P}(\hat{W}(k - 1) = i).$$

But  $\mathbb{P}(\hat{W}(k - 1) = i) = \mathbb{P}(W(k - 1) = i)$ , so  $\hat{W}(k) \stackrel{d}{=} W(k)$ , and similarly  $\hat{Y}(k) \stackrel{d}{=} Y(k)$ . By our assumption of  $Z_{k-1}(\cdot)$  and  $\tilde{Z}_{k-1}(\cdot)$ , for all  $i, j, \ell$ ,

$$\begin{aligned} &\mathbb{P}(\hat{W}(k) - i > \ell \mid \hat{W}(k - 1) = i, \hat{Y}(k - 1) = j) \\ &\geq \mathbb{P}(\hat{Y}(k) - j > \ell \mid \hat{W}(k - 1) = i, \hat{Y}(k - 1) = j), \end{aligned}$$

implying

$$\begin{aligned} & \mathbb{P}(\hat{W}(k) > \ell + i \mid \hat{W}(k-1) = i, \hat{Y}(k-1) = j) \\ & \geq \mathbb{P}(\hat{Y}(k) > \ell + i \mid \hat{W}(k-1) = i, \hat{Y}(k-1) = j) \end{aligned}$$

for all  $(i, j)$  such that  $i \geq j$ . (Note that  $\{\hat{Y}(k) > \ell + i\} \Rightarrow \{\hat{Y}(k) > \ell + j\}$  for  $i \geq j$ .) Multiplying both sides by  $\mathbb{P}(\hat{W}(k-1) = i, \hat{Y}(k-1) = j)$  and summing over all  $i, j$  such that  $i \geq j$ , and noting that  $\mathbb{P}(\hat{W}(k-1) \geq \hat{Y}(k-1)) = 1$ , we get

$$\mathbb{P}(\hat{W}(k) > \ell) \geq \mathbb{P}(\hat{Y}(k) > \ell) \quad \forall \ell \in \mathbb{Z}_+, \text{ i.e., } \hat{Y}(k) \leq_{st} \hat{W}(k).$$

$$\text{Since} \quad \hat{Y}(k) \stackrel{d}{=} Y(k) \quad \text{and} \quad \hat{W}(k) \stackrel{d}{=} W(k),$$

we get  $Y(k) \leq_{st} W(k)$ , completing the proof of stochastic dominance by induction. Since the limiting distributions of  $Y(t)$  and  $W(t)$  are the stationary distributions  $\mu_Y(\cdot)$  and  $\mu_W(\cdot)$  respectively (the convergence to the stationary distribution follows from [5], Chapter 4, Theorem 2.1), for random variables  $Y, W$  which are distributed according to  $Y \sim \mu_Y(\cdot)$  and  $W \sim \mu_W(\cdot)$ , we have  $Y \leq_{st} W$ . Hence the proof of the theorem is complete.

## C.8 Proof of Theorem 4.5

Consider the Lyapunov function  $Lyap(x) = x$ . For  $n$  large enough,

$$\mathbb{E}(W^{(n)}(t+1) - W^{(n)}(t) \mid W^{(n)}(t), W^{(n)}(t) > 0) \leq F^2 \eta e^{-cn} - \frac{1}{2} < -\frac{1}{3},$$

so the Lyapunov function has a negative drift outside the set  $\{0\}$ , and the positive recurrence of the Markov chain  $W^{(n)}(t)$  follows from Foster's theorem.

Further, the Markov chain is irreducible and aperiodic, so by Theorem 3.1 in [5], Chapter 3, the stationary distribution is unique.

We choose  $n$  large enough so that  $\eta e^{-cn} \leq 1$ . Let  $\pi_m = \mathbb{P}(W^{(n)}(0) = m)$  be the stationary distribution of  $W^{(n)}(t)$ , where we drop the explicit dependence on  $n$  for simplicity. We prove the following statement about  $\pi_m$  by induction:  $g(m) : \pi_m \leq \pi_0 e^{2\eta m} 5^{Fm} e^{-cnm}$ .

Base Case:

Clearly,  $g(0)$  is true, since  $\pi_0 = \pi_0$ .

Induction Step:

Let  $g(0), g(1), \dots, g(m-1)$  be true, and we need to prove  $g(m)$ . From the flow-balance equations for the Markov chain, and noting that  $\pi_j = 0$  for  $j < 0$ , we have

$$\pi_m \cdot \frac{1}{2} = \pi_{m-1} \sum_{j=1}^F \eta e^{-cnj} + \pi_{m-2} \sum_{j=2}^F \eta e^{-cnj} + \dots + \pi_{m-F} \eta e^{-cnF}.$$

Thus,

$$\begin{aligned} \pi_m &= 2\eta \sum_{r=1}^F \left( \pi_{m-r} \sum_{j=r}^F e^{-cnj} \right) \\ &\leq 2\eta F \sum_{r=1}^F \pi_{m-r} e^{-cnr} \\ &\stackrel{(a)}{\leq} 2\eta F \sum_{r=1}^F \pi_0 e^{2\eta(m-r)} 5^{F(m-r)} e^{-cn(m-r)} e^{-cnr} \\ &\leq 2\eta F^2 \pi_0 e^{-cnm} e^{2\eta(m-1)} 5^{F(m-1)} \\ &\leq \pi_0 e^{2\eta m} 5^{Fm} e^{-cnm}. \end{aligned}$$

Here, the step (a) follows from the induction hypothesis, and the last inequality follows from the following two relations:

1.  $2\eta e^{2\eta(m-1)} \leq e^{2\eta m}$ , since  $2\eta \leq e^{2\eta}$  for  $\eta > 0$ .
2. For  $x \geq 0$ , we have  $x^2 \leq 5^x$ , implying  $F^2 5^{F(m-1)} \leq 5^{Fm}$ , which holds since  $F$  is a positive integer.

Hence, by the principle of mathematical induction, the statement  $g(m)$  is true for all integers  $m \geq 0$ . It follows that for any integer  $s \geq 0$  and for  $n$  large enough, we have

$$\begin{aligned}
\mathbb{P}(W^{(n)}(0) > s) &= \pi_{s+1} + \pi_{s+2} + \cdots \\
&\leq \pi_0 \sum_{m=s+1}^{\infty} (e^{2\eta} 5^F e^{-cn})^m \\
&= \pi_0 \frac{e^{2\eta(s+1)} 5^{F(s+1)} e^{-cn(s+1)}}{1 - e^{2\eta} 5^F e^{-cn}} \\
&\leq 2\pi_0 e^{2\eta(s+1)} 5^{F(s+1)} e^{-cn(s+1)},
\end{aligned}$$

since  $e^{2\eta} 5^F e^{-cn} \leq 1/2$  for  $n$  large enough. Since  $\pi_0 \leq 1$ , we get

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}(W^{(n)}(0) > s) \geq c(s+1),$$

and the proof is complete.

## C.9 Proof of Lemma 4.4

Fix any  $i \in \{1, 2, \dots, m\}$ . Let the set of servers connected to  $Q_i$  be  $\mathcal{S}^* = \{S_1^i, S_2^i, \dots, S_{x_m}^i\}$ , in the increasing order of server-indices.

**Case 1:** The queue  $Q_i$  is allocated  $x$  or more servers.

In this case, because each server drains  $K$  packets, we have  $Q_i(t) \leq (L_i - xK)^+$ , implying

$$Q_i(t) \leq (L_i - xK)^+ \leq \max_{1 \leq j \leq m} (L_j - xK)^+,$$

and the claim holds.

**Case 2:** The queue  $Q_i$  is allocated at most  $x - 1$  servers.

In this case, by the pigeonhole principle, at least one of the queues  $Q_j \neq Q_i$  is allocated  $x + 1$  (or more) servers from the set  $\mathcal{S}^*$ . Consider the round  $y$  of SSG when the  $(x + 1)^{th}$  server from the set  $\mathcal{S}^*$  is allocated to  $Q_j$ . Since the SSG rule allocates a server to a longest queue among the queues it can serve, we have

$$Q_i^{(y-1)}(t) \leq Q_j^{(y-1)}(t) \leq (L_j - xK)^+,$$

and the result follows since  $Q_i(t) \leq Q_i^{(y-1)}(t)$ .

## C.10 Proof of Lemma 4.5

Let  $\zeta(t - 1) = c$ . By the choice of  $\epsilon$ ,

$$[p'_0, p'_1, \dots, p'_M] = [p_0 - M\epsilon, p_1 + \epsilon, p_2 + \epsilon, \dots, p_M + \epsilon]$$

is a probability vector with strictly positive components, and  $\sum_{i=1}^M p'_i \lceil i/K \rceil < 1$ . Further, since the set  $\mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}$  is compact and the

function  $g(\mathbf{z}) := \sum_{m=0}^M z_i \log \frac{z_i}{p_i}$  is lower semicontinuous ([7], Chapter 2, Exercise 2.1.22), the infimum of  $g(\mathbf{z})$  (in the definition of  $\tau$ ) is achieved and is strictly positive, since  $g([p_0, p_1, \dots, p_M]) = 0$  and  $g(\mathbf{z}) > 0$  for all other values of  $\mathbf{z}$ , and

$$[p_0, p_1, \dots, p_M] \notin \mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}$$

After arrivals in timeslot  $t$ , let  $\mathcal{F}_m$  be the set of queues whose length is at least  $c + m$ , and define  $f_m := |\mathcal{F}_m|/n$ . Consider the event

$$E := \left\{ f_m \leq \sum_{i=m}^M p'_i \quad \forall m \in \{1, 2, \dots, M\} \right\}.$$

From Sanov's theorem, we know that for  $n$  large enough,  $\mathbb{P}(E^c) \leq e^{-n\tau(1-\rho)}$  for any fixed  $\rho > 0$ . We condition the rest of this proof on the event  $E$ .

**Claim C.1.** *Conditioned on the event  $E$ , fix any integer  $m \geq 1$ , and any constant*

$$\delta \in \left( 0, q \left( 1 - \sum_{i=1}^M p'_i \left\lceil \frac{i}{K} \right\rceil \right) / (M(2-q)) \right).$$

*Then after  $n \left( \sum_{j=0}^{\infty} f_{m+jK} + (M-m+1)\delta \left( \frac{2}{q} - 1 \right) \right)$  rounds of server allocation under the SSG rule, with probability at least*

$$1 - Mn(1-q)^{n\delta} - Mn\delta e^{-\frac{2n\delta}{q} H(\frac{q}{2}|q)},$$

*there remain no queues of (updated) length  $c + m$  or more.*

Note that only a finite number of terms in the above infinite summation are nonzero.

*Proof.* Let  $\mathcal{F}_m^{(i)}$  denote the updated set  $\mathcal{F}_m$  after  $i$  rounds of server allocation, that is, the set of queues at length  $c + m$  or more, after  $i$  rounds of server allocation. First we consider the case  $m = M$ .

**Case 1:**  $|\mathcal{F}_M| = |\mathcal{F}_M^{(0)}| > n\delta$ .

For  $1 \leq i \leq m_0 := |\mathcal{F}_M^{(0)}| - n\delta$ , consider the event

$$W_i = \{X_{ri}(t) < K \quad \forall Q_r \in \mathcal{F}_M^{(i-1)}\}.$$

By the independence of channel allocation decision and the channel realizations for the higher-indexed servers,

$$\mathbb{P}(W_i | W_1^c, W_2^c, \dots, W_{i-1}^c) = (1 - q)^{(|\mathcal{F}_M^{(0)}| - i + 1)} \leq (1 - q)^{n\delta}.$$

If the server  $S_i$  is connected to any of the queues in the set  $\mathcal{F}_M^{(i-1)}$ , then (because the set  $\mathcal{F}_M^{(i-1)}$  is the set of the “current” longest queues) the server  $S_i$  is allocated to a queue in  $\mathcal{F}_M^{(i-1)}$ , and the served queue is removed from the set  $\mathcal{F}_M^{(i-1)}$  to obtain the set  $\mathcal{F}_M^{(i)}$ . Therefore, using the union bound, with probability at least  $1 - (|\mathcal{F}_M^{(0)}| - n\delta)(1 - q)^{n\delta} \geq 1 - n(1 - q)^{n\delta}$ , at the end of  $|\mathcal{F}_M^{(0)}| - n\delta$  rounds of service, we have  $|\mathcal{F}_M^{(m_0)}| \leq n\delta$ .

Consider the set of servers

$$\mathcal{S}^* = \{S_{m_0+1}, S_{m_0+2}, \dots, S_{m_0+2n\delta/q}\}.$$

By the Chernoff bound, the probability that a given queue  $Q_i \in \mathcal{F}_M^{(m_0)}$  is connected

- with a channel of rate =  $K$

- to at least  $n\delta$  of the servers in the set  $S^*$

is at least  $1 - e^{-\frac{2n\delta}{q}H(\frac{q}{2}|q)}$ . Therefore, by Lemma 4.4, at the end of  $2n\delta/q$  further rounds of service, with probability at least  $1 - n\delta e^{-\frac{2n\delta}{q}H(\frac{q}{2}|q)}$ , the maximum queue-length in the system is no more than  $c + m - 1$ , completing the proof for this case by the union bound.

**Case 2:**  $|\mathcal{F}_M| = |\mathcal{F}_M^{(0)}| \leq n\delta$ .

Following the analysis for case 1, it is clear that the claim holds in this case, completing the proof of the claim for the case  $m = M$ .

The proof of the claim now follows by repeatedly applying the above procedure to the cases  $m = M - 1, M - 2, \dots, 1, 0$ , using the union bound, and noting that if a queue  $Q_i$  of length  $\ell$  is served by a server  $S_j$ , then its length decreases by exactly  $K$  (or becomes zero), so that it continues to belong to the sets  $\mathcal{F}_{\ell-K}^{(j)}, \mathcal{F}_{\ell-K-1}^{(j)}$ , and so on.  $\square$

Applying the result of the claim to the case  $m = 1$ , we get

$$\begin{aligned}
& \sum_{j=0}^{\infty} f_{1+jK} + M\delta \left( \frac{2}{q} - 1 \right) \\
&= f_1 + f_{K+1} + \dots + f_{(r-1)K+1} + M\delta \left( \frac{2}{q} - 1 \right) \\
&\stackrel{(a)}{\leq} (p'_1 + p'_2 + \dots + p'_K) + 2(p'_{K+1} + \dots + p'_{2K}) + \dots \\
&\quad + r(p'_{(r-1)K+1} + \dots + p'_M) + M\delta \left( \frac{2}{q} - 1 \right) \\
&= \sum_{m=1}^M p'_m \left\lceil \frac{m}{K} \right\rceil + M\delta \left( \frac{2}{q} - 1 \right) \\
&< 1,
\end{aligned}$$

where the inequality (a) holds from Sanov's theorem (event  $E$  as defined above) with probability at least  $1 - e^{-n\tau(1-\rho)}$ , and the last inequality holds by the choice of  $\delta$ . Hence, by the union bound, we have (for  $n$  large enough)

$$\begin{aligned} \mathbb{P}(\zeta(t) > \zeta(t-1)) &\leq e^{-n\tau(1-\rho)} + Mn(1-q)^{n\delta} \\ &\quad + Mn\delta \exp\left(-\frac{2n\delta}{q} H\left(\frac{q}{2} \mid q\right)\right), \end{aligned}$$

completing the proof.

### C.11 Proof of Theorem 4.6

This proof proceeds in three steps, where we use Theorems 4.3, 4.4 and 4.5 to arrive at the desired conclusion.

**Step 1:** We have

$$\mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) > b\right) = \sum_{k=b+1}^{\infty} \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(0) = k\right).$$

Our aim is to calculate an upper-bound on the RHS of the above inequality. To this end, we know from Theorem 4.3 that  $\mathbb{P}(\max_{1 \leq i \leq n} Q_i(0) = k)$  is the same as  $\mathbb{P}(Y(t) = k)$ , where the (one-dimensional) Markov chain has the following transition probability structure:

$$\begin{aligned} \mathbb{P}(Y(t+1) = y \mid Y(t) = x) \\ = \mathbb{P}\left(\max_{1 \leq i \leq n} Q_i(t+1) = y \mid \max_{1 \leq i \leq n} Q_i(t) = x\right). \end{aligned}$$

**Step 2:** As a result of Lemma 4.6, we know that there exists a constant integer  $k_0$  such that over  $k_0$  consecutive timeslots and for  $n$  large enough, the

probability that the maximum queue-length in the system makes a transition from a higher value  $k$  to a lower value  $j$  (provided  $k > 0$ ) is at least 0.5.

Fix any constant  $\rho > 0$ . From Lemma 4.5, for  $n$  large enough, in a given timeslot, the probability that the maximum queue-length increases is at most

$$e^{-n\tau(1-\rho)} + Mn(1-q)^{n\delta} + Mn\delta \exp\left(-\frac{2n\delta}{q} H\left(\frac{q}{2} \mid q\right)\right).$$

Let

$$c := \min\left(\tau(1-\rho), \delta \log \frac{1}{1-q}, \frac{2\delta H\left(\frac{q}{2} \mid q\right)}{q}\right).$$

Fix any  $\epsilon' \in (0, c)$  and let  $c' = c - \epsilon'$ . For any given  $\epsilon > 0$  there exists  $n_0$  large enough such that for all  $n \geq n_0$ , we have

$$e^{-n\tau(1-\rho)} + Mn(1-q)^{n\delta} + Mn\delta \exp\left(-\frac{2n\delta}{q} H\left(\frac{q}{2} \mid q\right)\right) \leq e^{-nc'}.$$

Define  $L_i(t) := Q_i(k_0 t)$  and consider a Markov chain

$$L(t) = [L_1(t), L_2(t), \dots, L_n(t)].$$

The Markov chain  $L(t)$  is a time-sampled version of the Markov chain

$$[Q_1(t), Q_2(t), \dots, Q_n(t)],$$

and the two Markov chains have the same stationary distribution. Let  $L^*(t) = \max_{1 \leq i \leq n} L_i(t)$ . Define

$$p(k, j) := \mathbb{P}(L^*(t+1) = j \mid L^*(t) = k).$$

Then for any integer  $t$  and for any integer  $k > 0$ , we have  $\sum_{j=0}^{k-1} p(k, j) \geq 0.5$ .

Further, the probability that over a period of  $k_0$  consecutive timeslots the maximum queue-length increases in at least  $a \leq k_0$  timeslots is at most  $\binom{k_0}{a} e^{-anc'}$  by the union bound. More precisely, for  $1 \leq i \leq k_0$ , consider the events  $E_i := \{\zeta(i) > \zeta(i-1)\}$ . We know from Lemma 4.5 that regardless of the queue-lengths at the beginning of timeslot  $i$ , and any other events in any other timeslots,  $\mathbb{P}(E_i) \leq e^{-nc'}$  for  $n$  large enough. Hence,

$$\mathbb{P}\left(\bigcap_{i=1}^a E_i\right) = \prod_{i=1}^a \mathbb{P}(E_i \mid E_1, E_2, \dots, E_{i-1}) \leq e^{-anc'},$$

and the claimed bound follows from the union bound since there are  $\binom{k_0}{a}$  ways to “choose” the candidate timeslots where the maximum queue-length increases.

In a given timeslot, the length of any queue can increase by at most  $M$ , so that for any timeslot  $t$  if  $\zeta(t) = k$ , then  $\zeta(t+1) = s$  where  $s \leq k + M$ . It follows that for  $1 \leq a \leq k_0$ , we have

$$\sum_{j=(a-1)M+1}^{aM} p(k, k+j) \leq \binom{k_0}{a} e^{-anc'} \leq 2^{k_0} e^{-anc'}.$$

We now apply Theorem 4.4 with the random variables  $Z_t(k)$  and  $\tilde{Z}_t(k)$  with the following distributions:

$$\mathbb{P}(Z_t(k) = j) = p(k, k+j), \quad j \in \mathbb{Z},$$

for integers  $1 \leq a \leq k_0$  :

$$\mathbb{P}(\tilde{Z}_t(k) = j) = 2^{k_0} e^{-anc'}, \quad (a-1)M+1 \leq j \leq aM,$$

and  $\mathbb{P}(\tilde{Z}_t(k) = -1) = 0.5$ .

By the foregoing analysis, we have  $Z_t(i) \leq_{st} \tilde{Z}_t(j)$  for all  $i, j \in \mathbb{Z}_+$  and all  $t \in \mathbb{Z}$ , implying  $L^*(t) \leq_{st} W(t)$ . Further, the Markov chain  $W(t)$  has a stationary distribution for  $n$  large enough (follows from Theorem 4.5), and the Markov chain  $L(t)$  has a stationary distribution by the stability of the Markov chain  $Q(t)$  (follows from Theorem refthruput<sub>opt<sub>s</sub>sg</sub>). *Both the Markov chains  $W(t)$  and  $L(t)$  are clearlyaper* Hence, by Theorem 2.1, Chapter 4 in [5], the distributions of  $L^*(t)$  and  $W(t)$  converge to their respective stationary distributions, implying that the stationary distribution of  $L^*(t)$  is stochastically dominated by the stationary distribution of  $W(t)$ .

**Step 3:** We apply Theorem 4.5 with  $\eta = 2^{k_0}$  and  $F = k_0 M$  to conclude that

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P}(W(0) > b) = \liminf_{n \rightarrow \infty} \frac{-1}{n} \log \left( \sum_{k=b+1}^{\infty} \mathbb{P}(W(0) = k) \right) \geq \frac{c'(b+1)}{M},$$

implying

$$\begin{aligned} & \liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq n} Q_i(0) > b \right) \\ & \geq \frac{b+1}{M} \left[ \min \left( \tau(1-\rho), \delta \log \frac{1}{1-q}, \frac{2\delta H(\frac{q}{2}|q)}{q} \right) - \epsilon' \right] > 0, \end{aligned}$$

and completing the proof since the last inequality holds for all  $\epsilon' \in (0, c)$  for some  $c > 0$ , and for every constant  $\rho > 0$ .

## C.12 Proof of Lemma 4.7

Fix any  $j \in \{1, 2, \dots, n\}$ . We prove that the contribution of the server  $S_j$  to the weights of the schedules under the DMEQ rule and the MaxWeight

rule cannot differ by more than  $n\hat{\mu}^2$ . Once we prove this, the desired result follows by taking a summation over  $j = 1$  to  $n$ .

**Case 1:** If under the DMEQ rule  $\Lambda$ , the server  $S_j$  is allocated to  $Q_{b_j}$ , then the effective length of  $Q_{b_j}$  is at least  $\ell_{c_j} - n\hat{\mu}$ , because if not, then we can allocate  $S_j$  to  $Q_{c_j}$  (instead of  $Q_{b_j}$ ), which evidently has higher effective queue-length, and contradict the definition of the algorithm. (Informally, such an alternate allocation would result in “swapping” the server from a shorter queue to a longer (effective) queue.)

Thus, we have  $\ell_{b_j} \geq \ell_{c_j} - n\hat{\mu}$ . Further, from the Step 2 in the definition of the DMEQ rules (Definition 4.2), we have  $X_{b_j j} \geq X_{c_j j}$ , implying

$$\begin{aligned} \ell_{b_j} X_{b_j j}(t) &\geq (\ell_{c_j} - n\hat{\mu}) X_{c_j j}(t) \\ &\geq \ell_{c_j} X_{c_j j}(t) - n\hat{\mu}^2, \end{aligned}$$

where the last inequality holds because  $X_{ij}(t) \leq \hat{\mu}$  for all pairs  $(i, j)$  and all  $t$ .

**Case 2:** Consider the case when the DMEQ rule  $\Lambda$  does not allocate the server  $S_j$  to any queue. This can only happen if  $\ell_{c_j} \leq (n-1)\hat{\mu}$ , and if all the packets in the queue  $Q_{c_j}$  are served by the allocations under the rule  $\Lambda$ . For if not, then the server  $S_j$  can be allocated to  $Q_{c_j}$ , leading to a reduction in its (effective) queue-length and contradicting the definition of the algorithm. But, the maximum “weight” that the server  $S_j$  can contribute to any schedule is  $\ell_{c_j} X_{c_j j}(t)$ , by definition of  $c_j$ . Since  $\ell_{c_j} \leq n\hat{\mu}$  and  $X_{c_j j}(t) \leq \hat{\mu}$ , it follows that the server  $S_j$ ’s contribution to the weight of the MaxWeight schedule is at most  $n\hat{\mu}^2$ .

Thus the contribution of server  $S_j$  to the weights of the schedules under the DMEQ rule and the MaxWeight rule cannot differ by more than  $n\hat{\mu}^2$ , and the proof is complete.

### C.13 Proof of Lemma 4.8

Fix a node  $v \in \mathcal{V}$ . The probability that  $v$  has no edges incident on it is  $(1 - q)^n$ . Since this event implies that  $G$  has no  $r$ -fold perfect matching (w.r.t.  $\mathcal{U}$ ), the lower bound follows. For the upper bound, let  $\mathcal{U} = \bigcup_{i=1}^r \mathcal{U}_i$ , where each  $\mathcal{U}_i$  has exactly  $n$  nodes. Let  $G_i$  denote the bipartite graph obtained by restricting  $G$  to the set of nodes  $\mathcal{U}_i \cup \mathcal{V}$ . Since the sets  $\mathcal{U}_i$  are disjoint, the existence of perfect matchings in the graphs  $G_i$  are mutually independent events. By Lemma 2.1, for all  $1 \leq i \leq r$ , we have  $\mathbb{P}(G_i \text{ has no perfect matching}) \leq 3n(1 - q)^n$  for  $n$  large enough. Further, if  $\mathcal{M}_i$  is a perfect matching in  $G_i$ , then  $\bigcup_{i=1}^r \mathcal{M}_i$  is an  $r$ -fold perfect matching in  $G$  (w.r.t.  $\mathcal{U}$ ), and the upper bound is implied by the union bound.

### C.14 Proof of Lemma 4.9

Since the maximum possible channel rate is  $K$ , the DMEQ-rule  $\Lambda$  does not (additionally) set any channel rate  $X_{ij}(t)$  to 0 in step 2 of the Definition 4.2, if  $X_{ij}(t) = K$  at the beginning of the timeslot.

By hypothesis, there exists an  $w$ -fold perfect matching in  $G_w$ . Allocating servers to the queues as dictated by the matching would result in *each* of

the queues in the set  $\mathcal{C}_w$  getting allocated  $w$  servers, returning its queue-length to the value before arrivals in timeslot  $t$  or less, and the maximum effective queue-length at the end of timeslot  $T$  is no more than  $\zeta(t-1)$ . By definition, the maximum effective queue-length under the DMEQ-rule  $\Lambda$  is no more than *any* other allocation rule. It follows that  $\zeta(T) \leq \zeta(T-1)$ .

### C.15 Proof of Lemma 4.10

By the choice of  $\epsilon$ , we have

$$[p'_0, p'_1, \dots, p'_M] = [p_0 - M\epsilon, p_1 + \epsilon, p_2 + \epsilon, \dots, p_M + \epsilon]$$

is a probability vector with strictly positive components, and

$$\sum_{m=1}^M p'_m \left\lceil \frac{m}{K} \right\rceil < 1.$$

Further, since the set  $\mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}$  is compact and the function  $\sum_{m=0}^M z_i \log \frac{z_i}{p_i}$  is lower semicontinuous in  $z = [z_0, z_1, \dots, z_M]$  ([7], Chapter 2, Exercise 2.1.22), we know that the infimum is achieved and is strictly positive, since  $[p_0, p_1, \dots, p_M]$  is the only possible value of  $z$  for which the summation  $\sum_{m=0}^M z_i \log \frac{z_i}{p_i}$  is zero, and

$$[p_0, p_1, \dots, p_M] \notin \mathcal{M}_1(\Sigma) \setminus \{[p_0, p_1, \dots, p_M] + \mathcal{B}_\epsilon\}$$

Fix any  $\rho > 0$ . Let  $\mathcal{Q}_m$  denote the set of queues with exactly  $m$  arrivals. We have for all  $1 \leq m \leq M$ ,  $|\mathcal{Q}_m| \leq np'_m$  with probability at least  $1 - e^{-n\tau(1-\rho)}$ . Consider a partition of the set of servers  $\mathcal{S} = \bigcup_{m=0}^M \mathcal{S}_m$  where  $|\mathcal{S}_m| = n \left\lceil \frac{m}{K} \right\rceil p'_m$

for all  $m \geq 1$ .<sup>1</sup> Such a partition exists because  $\sum_{m=1}^M \left\lceil \frac{m}{K} \right\rceil p'_m < 1$  by the choice of  $\epsilon$ .

The bipartite graph defined by the set of nodes  $\mathcal{Q}_m \cup \mathcal{S}_m$ , where the edge realizations are defined by the channel realizations, there exists an  $\left\lceil \frac{m}{K} \right\rceil$ -fold matching with probability at least  $1 - 3np'_m \left\lceil \frac{m}{K} \right\rceil (1-q)^{np'_m} \geq 1 - 3n \left\lceil \frac{m}{K} \right\rceil (1-q)^{np'_m}$  (Lemma 4.8). Hence the proof is complete using the result of Lemma 4.9 and the union bound.

## C.16 Proof of Lemma 4.11

As mentioned, for ease of notation, we prove this claim only for the case  $p_M = p > 0$  and  $p_0 = 1 - p$ , and further assume that  $M = rK$  and  $q_0 = 1 - q$ .

By Assumption 4.1, we have  $pM < K$ . This together with  $M = rK$  implies  $p < 1/r$ . Fix a real number  $p' \in (p, 1/r)$ , say  $p' = (p + 1/r)/2$  for concreteness. By the Chernoff bound, the probability that in a given timeslot, more than  $np'$  of the queues see a nonzero (i.e.,  $M$ ) number of arrivals is at most  $e^{-nH(p'/p)}$ . Over a constant  $k_0$  number of timeslots, the probability that *at least one* of the timeslot has a nonzero number of arrivals to more than  $np'$  queues is at most  $k_0 e^{-nH(p'/p)}$ , by the union bound. Hence, if  $k_0$  is a constant independent of  $n$ , then the event of having at most  $np'$  queues with a nonzero number of arrivals in *each* of the  $k_0$  timeslots is a high probability

---

<sup>1</sup>We ignore the issues involving  $n \left\lceil \frac{m}{K} \right\rceil p'_m$  not being an integer, since they do not affect the proof in any essential way. The reader can instead consider  $|\mathcal{S}_m| = \left\lceil n \left\lceil \frac{m}{K} \right\rceil p'_m \right\rceil$ .

event, provided  $n$  is large. We condition the rest of the proof on this high probability event, and finally use the union bound to remove the conditioning. (Note that conditioning on a high probability is, at least in spirit, as good as no conditioning.)

Consider a fixed timeslot  $T \in \{t+1, t+2, \dots, t+k_0\}$ . In the timeslot  $T$ , the number of queues with a nonzero number of arrivals is at most  $np' < n/r$ . Let this set of queues be denoted by  $\mathcal{Q}^*$ . We know that  $|\mathcal{Q}^*| \leq np' < n/r$ . Consider the set of servers  $\mathcal{S}^* = \{S_1, S_2, \dots, S_{rnp'}\}$ , and let  $\mathcal{E}^*$  denote the set of edges corresponding to the channels of rate  $= K$  between the queues in  $\mathcal{Q}^*$  and the servers in  $\mathcal{S}^*$ . By Lemma 4.8, the probability that an  $r$ -fold perfect matching w.r.t  $\mathcal{Q}^*$  exists in the bipartite graph  $G^*(\mathcal{Q}^* \cup \mathcal{S}^*, \mathcal{E}^*)$  is at least  $1 - 3rnp'(1 - q)^{np'}$ . Note that allocating  $r$  servers to a queue with  $M = rK$  arrivals restores the queue-length to its value before arrivals, since each server drains  $K$  packets. Hence, if a candidate server allocation rule first makes allocation decisions for the servers in  $\mathcal{S}^*$  and then the rest (i.e., the servers in  $\mathcal{S} \setminus \mathcal{S}^*$ ), and allocates the servers in  $\mathcal{S}^*$  according to the  $r$ -fold perfect matching, then after the allocation of the servers in the set  $\mathcal{S}^*$ , we have:

1. All the queues are at an effective queue-length at most  $\zeta(T - 1)$ .
2. The number of servers left for allocation is  $|\mathcal{S} \setminus \mathcal{S}^*| = n(1 - rp')$ .

Further, the channel realizations for any two disjoint sets of servers are independent. It follows that it is possible to allocate the rest of the  $n(1 - rp')$  servers, one each to a queue with the maximum effective queue-length

(after the first step of allocating the servers in  $\mathcal{S}^*$ ), for  $n(1 - rp')$  queues at the maximum effective queue-length, which equals  $\zeta(T - 1)$  or less. Using Lemma 4.8, the probability of the existence of such an allocable matching is at least  $1 - 3n(1 - rp')(1 - q)^{n(1 - rp')}$ . Hence, we get either of the following with high probability:

1.  $\zeta(T) < \zeta(T - 1)$ .
2.  $\zeta(T) = \zeta(T - 1)$ , and  $\psi(T) \leq (\psi(T - 1) - n(1 - rp'))^+$ .

Since in any given timeslot, the DMEQ algorithm must have a smaller MEQ or a smaller number of queues at the MEQ as that of the candidate algorithm presented above, these bounds hold for the DMEQ algorithm as well.

Hence, by the union bound, for the DMEQ algorithm, for  $k_0 = \left\lceil \frac{1}{1 - rp'} \right\rceil$  timeslots, and with  $p' = (p + 1/r)/2$ , we have

$$\begin{aligned} \mathbb{P}(\zeta(t + k_0) \geq \zeta(t) \mid \zeta(t) > 0) &\leq \left\lceil \frac{1}{1 - rp'} \right\rceil \left[ e^{-nH(p'/p)} \right. \\ &\quad \left. + 3n \left( rp'(1 - q)^{np'} + (1 - rp')(1 - q)^{n(1 - rp')} \right) \right]. \end{aligned}$$

Since the RHS of the above inequality tends to zero as  $n$  tends to infinity, it is less than  $1/2$  for  $n$  large enough. Hence, with the choice  $p' = (p + 1/r)/2$ , we have the constant  $k_0 = \left\lceil \frac{1}{1 - rp'} \right\rceil = \left\lceil \frac{2}{1 - rp} \right\rceil$  such that for  $n$  large enough,

$$\mathbb{P}(\zeta(t + k_0) < \zeta(t) \mid \zeta(t) > 0) \geq \frac{1}{2},$$

which is the desired result.

## C.17 Proof of Lemma 4.12

The bipartite graph between the set of queues and the set of servers, defined by the channel realizations with rate  $= K$ , exhibits a perfect matching with probability at least  $1 - 3n(1 - q)^n$  for  $n$  large enough (follows from Lemma 4.8). If such a matching exists, then it is possible to allocate one server to each queue, and since the number of arrivals to a queue in a given timeslot is less than the channel rate, the length of any queue does not increase in the given timeslot. Since the DMEQ rule must result in a MEQ that is smaller than or equal to the MEQ under *any* other scheduling rule, the result follows.

## C.18 Proof of Theorem 4.12

**Throughput-optimality:** It can be shown (along the lines of the proofs of Lemmas 4.7 and Lemma *maxwt<sub>m</sub> inus<sub>c</sub> onst<sub>s</sub> sg*) that under the same queue lengths, channel realizations and arrivals at the beginning of a timeslot, the weight of the schedule under optimal *MaxWeight* rule is at most an additive constant more than that under the DMEQ and the SSG. Thus, the proof of throughput-optimality of the DMEQ and SSG algorithms (under Assumption 3.1) follows the proofs of Theorems 4.8 and 3.5.

**Rate-function:** Consider a symmetric system with  $\xi n$  queues and  $\xi n$  servers. For this system,

$$\liminf_{n \rightarrow \infty} \frac{-1}{n} \log \mathbb{P} \left( \max_{1 \leq i \leq \xi n} Q_i(0) > b \right) \geq \xi \Gamma,$$

since

$$\liminf_{n \rightarrow \infty} \frac{-1}{\xi n} \log \mathbb{P} \left( \max_{1 \leq i \leq \xi n} Q_i(0) > b \right) \geq \Gamma.$$

Given a system with  $n$  queues and  $\xi n$  servers, we add  $(\xi - 1)n$  “dummy” queues and analyze the symmetric system, for which the analysis so far (in Sections 4.4 and 3.3) is valid. The transition probability bounds for the Markov chain for the symmetric system (Theorem 4.9) clearly hold for a system with more servers than queues, and so do the rate-function bounds. In other words, the symmetric system with the “dummy” queues provides a lower bound on the performance of the asymmetric system with  $n$  queues and  $\xi n$  servers, thus completing the proof.

## Bibliography

- [1] 3GPP TR 25.913. Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN). March 2006.
- [2] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, and P. Whiting. CDMA data QoS scheduling on the forward link with variable channel conditions. *Bell Labs Tech. Memo*, April 2000.
- [3] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Scheduling in Multi-Channel Wireless Networks: Rate Function Optimality in the Small-Buffer Regime. In *Proc. SIGMETRICS/Performance Conf.*, Jun. 2009.
- [4] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Low-complexity Scheduling Algorithms for Multi-channel Downlink Wireless Networks. In *Proc. IEEE Infocom*, Mar. 2010.
- [5] Pierre Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer Science + Business Media, LLC, 2008.
- [6] Loc Bui, R. Srikant, and Alexander Stolyar. Novel Architectures and Algorithms for Delay Reduction in Back-Pressure Scheduling and Routing. In *Proc. IEEE Infocom*, Apr. 2009 (to appear).
- [7] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer-Verlag New York, Inc., second edition, 1998.

- [8] Rick Durrett. *Probability: Theory and Examples*. Brooks-Cole, Thomson Learning, 3rd edition, 2005.
- [9] J. Edmonds. Maximum Matching and a Polyhedron with  $(0, 1)$  vertices. *J. Res. Natl. Bureau Standards*, 69(B):125–130, 1965.
- [10] A. Eryilmaz, R. Srikant, and J. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Trans. Network.*, 13:411–424, April 2005.
- [11] WiMax Forum. Mobile WiMAX Part I: A technical overview and performance evaluation. March 2006. White Paper.
- [12] A. El Gammal, J. Mammen, B. Prabhakar, and D. Shah. Optimal throughput-delay scaling in wireless networks - part I: The fluid model. *IEEE Trans. Inform. Theory*, 52(6):2568–2592, June 2006.
- [13] A. El Gammal, J. Mammen, B. Prabhakar, and D. Shah. Optimal throughput-delay scaling in wireless networks - part II: Constant-size packets. *IEEE Trans. Inform. Theory*, 52(11):5111–5116, November 2006.
- [14] A. Ganti, E. Modiano, and J. N. Tsitsiklis. Optimal Transmission Scheduling in Symmetric Communication Models With Intermittent Connectivity. *IEEE Trans. Inform. Theory*, 53(3):998–1008, Mar. 2007.

- [15] P. Gupta and T. Javidi. Towards Throughput and Delay-Optimal Routing for Wireless Ad-Hoc Networks. In *Asilomar Conference on Signals, Systems and Computers*, Nov. 2007.
- [16] John E. Hopcroft and Richard M. Karp. An  $n^{5/2}$  Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM Journal on Computing*, 2(4):235–231, Dec. 1973.
- [17] S. Jagabathula and D. Shah. Optimal Delay Scheduling in Networks with Arbitrary Constraints. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Jun. 2008.
- [18] S. Kittipiyakul and T. Javidi. Delay-Optimal Server Allocation in Multiqueue Multiserver Systems with Time-Varying Connectivities. *IEEE Trans. Inform. Theory*, 55(5):2319 – 2333, May 2009.
- [19] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Pearson Education, 2006.
- [20] P. R. Kumar and Sean P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Trans. Automat. Contr.*, 40:251–260, Feb. 1995.
- [21] S.P. Meyn. Stability and asymptotic optimality of generalized maxweight policies. *SIAM J. Control and Optimization*, 2008. to appear.
- [22] M. J. Neely. Optimal energy and delay tradeoffs for multiuser wireless downlinks. *IEEE Trans. Inform. Theory*, 53(9):3095–3113, Sept. 2007.

- [23] M. J. Neely. Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks. *IEEE/ACM Trans. Network.*, 16(5):1188–1199, 2008.
- [24] M. J. Neely, E. Modiano, and C. E. Rohrs. Power and server allocation in a multi-beam satellite with time varying channels. In *Proc. IEEE Infocom*, volume 3, pages 1451–1460, New York, NY, June 2002.
- [25] S. Shakkottai. Effective capacity and QoS for wireless scheduling. *IEEE Trans. Automat. Contr.*, 53(3):749–761, February 2008.
- [26] S. Shakkottai, R. Srikant, and A. Stolyar. Pathwise optimality of the exponential scheduling rule for wireless channels. *Ann. Appl. Prob.*, 36(4):1021–1045, December 2004.
- [27] S. Shakkottai and A. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Ann. Math. Statist.*, 207:185–202, 2002.
- [28] A. Stolyar. MaxWeight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *Ann. Appl. Prob.*, 14(1), 2004.
- [29] A. Stolyar. Large deviations of queues sharing a randomly time-varying server. *Queueing Systems*, 59:1–35, 2008.
- [30] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in

multihop radio networks. *IEEE Trans. Automat. Contr.*, 4:1936–1948, December 1992.

- [31] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. Inform. Theory*, 39:466–478, March 1993.
- [32] V. J. Venkataramanan and X. Lin. Structural properties of LDP for queue-length based wireless scheduling algorithms. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, Monticello, Illinois, September 2007.
- [33] L. Ying, R. Srikant, A. Eryilmaz, and G. Dullerud. A large deviations analysis of scheduling in wireless networks. *IEEE Trans. Inform. Theory*, 52(11):5088–5098, November 2006.
- [34] Lei Ying, Sanjay Shakkottai, and Aneesh Reddy. On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks. In *Proc. IEEE Infocom*, Apr. 2009.

# Index

Abstract, vii  
*Acknowledgments*, v  
*Appendices*, 100  
  
*Dedication*, iv  
DeepestDrain Operation, 82  
DMEQ Rule, 84  
Dominance property, iLQF, 22  
Drain property, 24  
  
iHLQF Algorithms, 46  
iLQF Algorithms, 19  
iLQF with PullUp, 27  
  
MaxWeight Algorithm, 52  
Modified iLQF with PullUp, 50  
  
Perfect-matching scheduling, 18  
PullUp operation, 25  
  
r-fold perfect matching, 87  
  
SSG Rule, 54

## Vita

Shreeshankar Bodas is a graduate student in The Department of Electrical and Computer Engineering at The University of Texas at Austin. He graduated from IIT Madras, India in 2005 with B. Tech. in Electrical Engineering, and received his M.S.E. degree in Electrical and Computer Engineering from UT Austin in 2007. He worked as a summer intern at Qualcomm, Inc. (2008) and Samsung Telecommunications America (2010). His research interests include queuing theory and scheduling algorithms for communication networks.

Permanent address: 284 Harvard Street  
Apt 66  
Cambridge, MA 02139

This dissertation was typeset with  $\text{\LaTeX}^\dagger$  by the author.

---

<sup>†</sup> $\text{\LaTeX}$  is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's  $\text{\TeX}$  Program.