View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UT Digital Repository

Copyright

by

ChangWoo Yang

2007

The Dissertation Committee for ChangWoo Yang
certifies that this is the approved version of the following dissertation:

Large Deviations Analysis of Scheduling Policies for a Web Server

Committee:

Sanjay Shakkottai, Supervisor

Ari Arapostathis

Vijay Garg

John Hasenbein

Srinidhi Varadarajan

Gustavo de Veciana

Large Deviations Analysis of Scheduling Policies for a Web Server

by

ChangWoo Yang, B.S.E.;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 2007

With love,

to my family

Acknowledgments

I would like to express my deepest gratitude and appreciation to Prof. Sanjay Shakkottai for his guidance, support, patience, and encouragement throughout this long journey. It was his valuable scholarly advice and generous support that helped the most to finish my PhD. I feel very fortunate and extremely proud to have worked with him.

I am indebted to my dissertation committee members, Prof. Ari Arapostathis, Prof. Vijay Garg, Prof. John Hasenbein, Prof. Srinidhi Varadarajan, and Prof. Gustavo de Veciana for their insightful and constructive comments that helped me to improve the quality of my work to a great extent. In addition, I would like to extend my sincere thanks to Prof. JaiYong Lee at the Yonsei University for his warm advice and constant support.

Most of all, I am extremely grateful to my parents, ByungHo Yang and YoungHee Park. They have been a source of unconditional love, encouragement, and belief for which I would not have reached so far. Thank you to my parent in law for their many prayers and concern for my well-being. Also, my thanks goes to old friends in Korea and new friends that I have made during my graduate studies. Lastly, I express my deepest thanks to my wife YoungIhn Koh for her heart warming encouragement, love and support. I love you with all my heart and soul.

Large Deviations Analysis of Scheduling Policies for a Web Server

Publication No. ______________________

ChangWoo Yang, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Sanjay Shakkottai

With increasing demand and availability of bandwidth resources, there has been tremendous growth in the scale and speed of web servers. In web servers, scheduling plays an important role in resource allocation (for instance, bandwidth allocation, processor allocation, etc). However, as the scale of a system increases so does the number of activities/events in the system (e.g., job arrivals), as a consequence of which the analysis of scheduling becomes increasingly harder. In particular, the possible ways in which scheduling failure (e.g., queue overflow, excessively large delay, instability of a system) can occur becomes increasingly greater, thus making it more difficult to understand the behavior and develop design rules for scheduling algorithms. However, a well-known observation from large deviations theory that large scale systems fails in a “most likely way” can potentially be used to simplify the design and analysis of scheduling. In this thesis, we study the implications and applications of this effect on scheduling in a web server accessed by a large number of sources.

We analyze the delay distribution of scheduling policies for web servers under a many sources large deviation regime which models web servers in a large scale system well. Due to the difficulties brought on considering a large number of sources, only a small number of scheduling policies, such as First-Come-First-Serve (FCFS), General-Processor-Sharing (GPS), and Priority Queueing policies have been analyzed under the many sources regime. In particular, in a single queue single server setup the delay characteristics of only FCFS, Shortest-Job-First (SJF), and Longest-Job-First (LJF) has been analyzed.

In this thesis, we study the Two-Dimensional-Queueing (2DQ) framework, a unifying queueing framework that allows the identification of the “most likely way” in which delay occurs, to analyze the delay of various unexplored scheduling policies. In conjunction with the 2DQ framework, we develop a new “cycle based” technique for understanding the large deviations tail probability of more complex policies.

Using the combination of the 2DQ framework and the cycle based analysis, we first analyze two interesting scheduling policies, i.e., Shortest-Remaining-Processing-Time (SRPT) policy (which is mean delay optimal) and Processer-Sharing (PS) policy (which is a “fair” policy). We derive the asymptotic delay distributions (rate functions) of both policies and study their behavior across job sizes. Next, we address three problems in implementing the aforementioned scheduling policies: (i) end receivers may have bandwidth constraints that are not taken account in SRPT, (ii) the remaining processing time information might not be available to the web-server, and (iii) most actual implementations are variants of SRPT to reflect other implementation constraints and/or to jointly optimize other metrics in addition to delay, i.e., jitter, fairness, etc. To address these, we first develop finite-SRPT that takes into account the bandwidth constraint at the end receiver, and show that the policy shifts between SRPT and a PS-like policy depending on the bandwidth constraint. Second, we study the Least-Attained-Service (LAS) policy which is viewed as a good substitute for SRPT when the remaining job size is not available and we analyze the penalty associated with not using the remaining size information directly. Lastly, we analyze a class of scheduling policies known as SMART that contains many variants of SRPT with different fairness properties and show that all policies in the class have the same tail probability of delay across job sizes for a many sources regime. The results of this thesis facilitate the understanding of various scheduling policies under the many sources regime and provides an analytical queueing framework that can be used to understand other scheduling policies.

Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **xii**

List of Figures **xiii**

Chapter 1 Introduction **1**

1.1 Summary 1

1.2 Main Contributions 6

1.3 Organization 7

Chapter 2 The Basic System Model **8**

2.1 The System Setup 9

2.2 Many–Sources Large–Deviations Regime 10

2.3 Brief Overview of Scheduling Policies 13

Chapter 3 SRPT **15**

3.1 Background and Related Work 15

3.2 Main Contributions and Intuition 17

3.3 Numerical Analysis 23

Chapter 4 SMART **25**

4.1 Background and Related Work 25
4.2 Two Dimensional Queueing Framework 28
4.3 Intuition 31
4.4 Main Contributions 32
    4.4.1 SMART for Large Deviations 32
    4.4.2 Analytical Result 33
4.5 Numerical Analysis 38

Chapter 5 LAS **42**

5.1 Background and Related Work 42
5.2 Intuition 44
5.3 Main Contributions 46
5.4 Numerical Analysis 50

Chapter 6 finite-SRPT **54**

6.1 Background and Related Work 54
    6.1.1 Definition and explanation of finite-SRPT 55
6.2 Intuition 56
6.3 Main Contributions 59
    6.3.1 Characteristics of finite-SRPT 59
    6.3.2 The delay decay rate of finite-SRPT 61
6.4 Numerical Analysis 67

Chapter 7 Discrete Processor Sharing **69**

7.1 Background and Related Work 69
7.2 Two Dimensional Queueing Framework with cycles 71
7.3 Intuition 72
    7.3.1 PS in discrete time 72

7.3.2 Discrete PS described through 2DQ framework with cycles 73
7.4 Main Contributions 77
7.5 Numerical Analysis 83

Chapter 8 Concluding Remarks **85**

Appendix A Proof of Theorems **88**

A.1 Proof of Theorem 3.2.1 88
A.2 Proof of Theorem 3.2.2 92
A.3 Proof of Theorem 4.4.1 95
A.4 Proof of Theorem 5.3.1 101
A.4.1 Lower bound of the decay rate 101
A.4.2 The delay decay rate (a tight upper bound) 104
A.5 Proof of Theorem 6.3.1 107
A.5.1 Lower bound on the decay rate 107
A.5.2 Upper bound of the decay rate 113
A.6 Proof of Theorem 7.4.1 116
A.6.1 Lower bound of the delay decay rate of discrete PS 116
A.6.2 Tight upper bound of the delay decay rate of discrete PS 120

Bibliography **122**

Vita **127**

List of Tables

2.1 Notations frequently used in the dissertation. 11
2.2 Overview of scheduling policies. 14

List of Figures

2.1 Illustration of many sources framework. In the many sources large deviations framework, we consider the probability of large deviations from the mean when the system is accessed by a large number of sources and is served by a server with capacity that increase in proportion. 9

3.1 Illustration of the delay decay rate of size 1 and M jobs for SRPT compared with FIFO. The decay rate of size 1 jobs for SRPT is actually infinite since the setup ensures that all size 1 jobs are served as soon as they arrive. It is depicted as a horizontal line for viewing purpose. Note that the decay rate of size M jobs for SRPT approaches that of FIFO as the M increases while the difference for size 1 jobs remains large (actually grows for large M). . . 23

4.1 Illustration of the Two Dimensional Queueing (2DQ) Framework. The left figure depicts the basic form of the 2DQ framework which is basically a collection of virtual queues arranged in a grid to reflect the state of the system. The right figure depicts the 2DQ representation for SMART where the X-axis is the original size and the Y-axis is the remaining processing time of a job. 29

4.2 Illustrations of the two dimensional queueing framework for SMART. Note that since a job cannot have remaining size larger than its original size only the lower right triangle of Figure 4.1 is of importance. The progression of a job between queues while in the system is illustrated in (a). The priority structure for an incoming job is shown in (b) and for a partially served job is shown in (c). 30

4.3 Plot of the rate of convergence of SRPT, PSJF, and RS to the decay rate under the uniform workload with $M = 16$, $\rho = 0.8$, and $m = 4$. The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation since a large delay for smaller job sizes is a very low probability event as N grows. Though not shown here, we found similar convergence rates under other policies in SMART-LD. 38

4.4 Plot of the decay rate as a function of the threshold m under the power-law and the high variability workload under SMART-LD with the maximum job size $M = 16$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted. 39

4.5 Plot of the decay rate as a function of the maximum job size M under the power-law and the high variability workload under SMART-LD with the threshold $m = 4$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted. 40

5.1 Illustration of the 2DQ representation for LAS. As depicted the X-axis is the original size and the Y-axis is the attained service of a job, where attained service is the amount of service a job has received so far. 44

5.2 Illustrations of the two dimensional queueing framework for LAS. Note that since a job cannot have attained service larger than its original size only the upper right triangle of Figure 5.1 is of importance. The progression of a job between queues while in the system is illustrated in (a). The priority structure for an incoming job is shown in (b) and for a partially served job is shown in (c). 45

5.3 Plot of the rate of convergence of SRPT to the decay rate under the uniform workload with $M = 16$, $\rho = 0.8$, and $m = 4$. The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation since a large delay for smaller job sizes is a very low probability event as N grows. Though not shown here, we found similar convergence rates under other policies in SMART-LD. 49

5.4 Plot of the decay rate as a function of the threshold m under the power-law and the high variability workload under SMART-LD with the maximum job size $M = 16$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted. 50

5.5 Plot of the decay rate as a function of the maximum job size M under the power-law and the high variability workload under SMART-LD with the threshold $m = 4$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted. 51

5.6 Plot of the delay rate function as a function of the job size, k, with the threshold $m = 20$ and maximum job size $M = 16$ held fixed. Recall that $I_W(k, m)$ measures the rate function of $Pr(W(k) > m)$ and that a larger $I_W(k, m)$ indicates a stochastically smaller delay. 52

6.1 Finite-SRPT and two dimensional queueing framework. The left-most figure depicts the path that a job should take for it to be fully serviced. X-axis is the original size of a job and Y-axis is the remaining size of a job. Thus a job of original size k upon receiving service progresses downward until it is fully served, i.e., remaining processing time is 0. However, different sized jobs require different numbers of rounds of D units of service in finite-SRPT. jobs of size $k \leq D$ will be fully served in one round, while jobs with size $D < k \leq 2D$ require two rounds of service to leave the system. 56

6.2 Priority scheme for a tagged job of size k, where $D < M \leq 2D$. As depicted in the figures, all fictitious queues in the shaded area are of higher priority. Correspondingly, lower priority queues are the fictitious queues in the non-shaded area. The higher priority area is composed of typical higher priority jobs represented as the thick diagonal lines and atypical jobs are in the rest of the shaded area. 57

6.3 Finite-SRPT described with respect to SRPT and PS. Finite-SRPT occupies a region between SRPT and PS for varying values of D. When $D = M$, finite-SRPT is equivalent to SRPT. Finite-SRPT at $D = 1$ is a scheduling policy very close to PS but different in the fact that only jobs with smaller remaining processing time are served before the tagged job in a cycle instead of all jobs in the system. 60

6.4 Delay decay rate of job sizes $1, 2, 4, 6, 8$, for varying $D = \{2, 4, 8\}$, exceeding the threshold $m = 5$ is depicted. X-axis is the original size of the tagged job and y-axis is the actual numerics of the delay decay rate. 67

7.1 Illustration of the two queue representation of the discrete PS operation. The queue with the star arrow is the queue that is active, i.e., queue being served. A job receives a unit whence it is transferred to the other queue without the star arrow, i.e., the inactive queue. When all jobs leave the active queue, the other queue becomes active where the process repeats again. 72

7.2 Illustration of the infinite queue representation. The argument is given that the infinite queue representation is equivalent to the two queue representation of discrete PS. 73

7.3 Illustration of the 2DQ with cycles representation of the discrete PS scheduling policy. The 2DQ representation follows the two queue representation and provides a tractable framework that makes the analysis of discrete PS possible. 74

7.4 Illustration of the progress of a tagged size 3 job in the 2DQ representation. Note that the transition of active queues in the two queue representation is expressed as the sequence of multiple 2DQs with strict priority between them. 75

7.5 Illustration of the 2DQs representing the state of the system for a tagged job of size k after time slot 0. We denote $2DQ/0$ as the inactive 2DQ at time 0, i.e., the 2DQ that receives new arrivals at time 0. Active cycle denote the time interval that the respective 2DQ is active and the job arrivals are the actual arrivals that the particular 2DQ receives. Note that for size k tagged job only 2DQs ranging from $2DQ/0$ to $2DQ/k$ are relevant since the tagged job is fully served and leaves the system in $2DQ/k$. 76

7.6 Illustration of the 2DQs representing the state of the system for a tagged job of size k before time slot 0. The 2DQ number is assigned relative to $2DQ/0$. The last 2DQ corresponds to the 2DQ that receives job arrivals at the start of the busy period, where $T_* - 1$ is the last time before time slot 0 that the system was empty. 77

7.7 Illustration of the simplification of the past 2DQs. As depicted the unknown number of 2DQs after $2DQ/k - M$ can all be combined into a single 2DQ that represent the 2DQ with the corresponding job arrivals that need to be fully served before the tagged job leaves the system. This simplification is one of the key observations that make the analysis of discrete PS possible. . 81

7.8 Illustration of the multiple resolution of 2DQ framework for discrete PS with respect to cycles. The figure shows the simplification of the past 2DQs and the additional resolution of $2DQ/0$ where the priority scheme before and after time 0 is different. 82

7.9 Plot of the delay decay rate of discrete PS under the exponential distribution with $\rho = 0.8$, $m = 5$, and $C = 1$ for jobs sizes $1, 3$, and 5. The asymptotic decay rate of discrete PS shows that even the exponential decay rate of discrete PS is dependent on the job size where the decay rate decreases for larger jobs, i.e., slightly favors small jobs. However, compared to SRPT and LAS the discrete PS policy does not favor small jobs as much. The delay decay rate of SRPT and LAS for size 1 jobs are actually infinite, but depicted as finite to able to compare the delay decay rate of other job sizes. 83

A.1 Illustration of the bounds of θ^* that are used to prove Theorem A.2.1. As shown in the figure, we construct $f_k(\theta)$ that lower bounds $\Lambda_{A_k,0}(\theta)$ for all θ. 92

A.2 Illustration of the lower and upper bound of the actual delay with respect to the virtual delay. The lower bound shows that the actual job can leave the system only if the virtual job leaves the system, i.e., reaches the head of the queue. The upper bound shows that if the virtual job behind the additional size k job reaches the head of the queue then the actual job is guaranteed to be fully served. 95

A.3 Priority scheme for a tagged job of size k, where $D < M \leq 2D$. The higher priority area is composed of typical higher priority jobs represented as the thick diagonal lines and atypical jobs are in the rest of the shaded area that represent the atypical jobs. 110

Chapter 1

Introduction

1.1 Summary

With increasing demand and availability of bandwidth resources, there has been tremendous growth in the scale and speed of computer system networks. In such a large scale system, it is understood that various limited resources must be shared efficiently. Specifically, scheduling is a key consideration for efficient resource allocation. Thus, exploring numerous effects and applications of scheduling in large scale systems is a crucial step in understanding the current Internet.

Models used to understand computer systems tends to capture the properties of small to medium sized systems better than a large scale system. Extension of these models to large scale systems mostly entails approximations or omissions of important characteristics of large scale systems. Understanding scheduling under a model specifically tailored to capture the properties of large scale systems is important to precisely understand how and why scheduling policies behave as they do in current computer systems.

In this dissertation, we consider the many sources regime for the analysis and modeling of scheduling in web servers in a large scale system. The many sources regime models web servers in large scale systems well in that the regime portrays the increasing number of sources/flows evident in the Internet. In addition, to accommodate the huge amount traffic due to the increased number of sources, most service providers in turn have increased the capacity of their servers as well as their queue size. The many source regime captures this trend well by scaling the number of arrival flows along with the capacity and the queue size. Thus, analysis of scheduling policies under the many sources regime provides a more accurate understanding of scheduling in current day web servers.

It is natural to expect that considering web servers in a large scale system would lead to increased complexity which would make analysis and modeling of scheduling difficult, which has certainly been the reason behind the scarcity of the analysis of scheduling in a many sources regime. However, we use the intuition that the sample path behavior of a large scale system can be characterized by a single "most–likely trajectory". In other words, as the number of sources scales, minor fluctuations away from the most-likely behavior of scheduling can be effectively ignored and as a result the main characteristics or modes of behavior sufficiently describes and models scheduling. Based on such intuition, we study the delay performance of scheduling policies in web servers accessed by a large number of sources.

Considering the emphasis on QoS in computer systems, an important delay performance to consider is the tail probability of delay. In this context, we are interested in the probability that the delay experienced by a user exceeds some threshold in a web server accessed by many sources. In particular, the decaying trend of the probability (rate function) of delay experienced by a typical job exceeding a threshold (Complementary Cumulative Distribution Function (CCDF) of delay) in a web server is derived under the many sources large deviations framework[1] [8, 10, 24]. Due to the complexity of large scale systems, only a few scheduling policies, such as First-In-First-Out (FIFO) [10], General-Processor-Sharing (GPS) [26], and Priority Queueing policies [13, 38] have been analyzed under the many

---

[1] The many sources large deviations framework basically states that the probability of rare event occurring (CCDF of delay in our case) decays exponentially with a specific exponent (the rate function) as the system becomes large (the many sources regime).

sources large deviations framework. This is due to the fact that as the scale of the system increases so does the number of activities/events in the system (e.g., job arrivals), as a consequence of which the possible ways in which a user experiences delay exceeding a threshold can occur becomes increasingly large, thus making it more difficult to understand the behavior and develop design rules for scheduling algorithms.

As a step toward completing the picture for delay analysis of scheduling policies, we study schedulers with a single server and a single queue in the many sources large deviations regime. In this context, only the delay distribution of FIFO [10], Shortest-Job-First (SJF), and Longest-Job-First (LJF)[2] have been reported in literature. Numerous other scheduling policies, such as schedulers that prioritize jobs with respect to the remaining processing time, and attained service of jobs, have escaped analysis due in large part to the increased complexity in large scale systems.

In this dissertation, we study the Two-Dimensional-Queueing (2DQ) framework [48], a novel unifying queueing framework that allows the analysis of delay experienced by a user for various scheduling policies in the many sources large deviations regime. The 2DQ framework is a collection virtual queues ordered in such a way that accurately describes the state of the system at all times. More importantly, the system of virtual queues allows consistent and coherent separation of higher and lower priority areas and in some cases no priority areas of virtual queues with respect to the job under consideration.

In conjunction with the 2DQ framework, we develop a new "cycle based" technique for understanding the large deviations tail probability of more complex policies. Such complex scheduling policies possess priority schemes that can not be completely described using a single 2DQ, i.e., 2DQ framework. We improve upon the 2DQ framework to multiple 2DQs, i.e., multiple collection of virtual queues, where each 2DQs are "active" in well defined time intervals (cycles). The multiple 2DQs facilitates analysis by capturing the additional prioritization in the more complex scheduling policies.

---

[2] The delay distribution of SJF and LJF can be derived by a simple application of the results on Priority Queueing scheduling policy [13, 38].

We first analyze two interesting scheduling policies, the Shortest-Remaining-Processing-Time (SRPT) policy through the 2DQ framework and the Processer-Sharing (PS) policy using 2DQ framework in conjunction with the cycle based analysis. SRPT is an interesting policy since it is optimal in delay performance, i.e., SRPT is shown to possess the smallest mean delay compared to any work conserving scheduling policy [37]. On the other hand, at the other end of the spectrum are scheduling policies that are fair. In the literature, PS has been regarded as *the* fair policy in various metrics, i.e., fair distribution of the capacity [22, 23], slowdown[3] [42], etc. By understanding these two scheduling policies that exemplify scheduling policies that possess good mean delay characteristics and that are fair in delay, we believe that a better grasp of other scheduling policies can be achieved. In particular, we derive the asymptotic (asymptotic in the number of sources) delay distributions of both scheduling policies and study their behavior across job sizes.

Next, we address the following three problems in implementing the aforementioned scheduling policies.

(i) SRPT services a single highest priority job until it is fully served and leaves the system or until a job of even smaller remaining processing time arrives to the system whence it will be preempted. Thus the ideal SRPT model assumes that a job receives the full capacity of the web-server during the duration of its service. However, one problem in implementing SRPT in practical systems is that SRPT does not take into consideration the bandwidth constraint at links. In other words, SRPT guarantees full bandwidth (BW) of the server to job requests with the smallest remaining processing time until the request are fully accommodated or are preempted. This is unrealistic when we consider servers of large BW.

(ii) The SRPT policy assumes that the remaining processing time information is available to the web-server. A web-server derives the remaining processing time information by receiving the original size information at the time a job arrives. However, in some

---

[3] Slowdown of a job is its delay divided by the size of that job.

cases the original size and hence the remaining processing time information might not be available to the web-server. For example, a request of a web page may entail downloading or searching other web sites for files of unknown size.

(iii) SRPT is an idealistic policy which in many cases is not implemented directly [28, 33, 34]. In fact, many practical implementations are variants that reflect implementation constraints and/or to jointly optimize other metrics in addition to delay, i.e., jitter, fairness, etc.

In the dissertation, we address the above three questions in the following manner.

(i) We develop the finite-SRPT policy that takes into account the bandwidth constraint at the end receiver, i.e., jobs are served at most a given fixed amount at any time. This reflects the practical BW constraint of the end users. We analyze the delay of finite-SRPT as the bandwidth constraint is varied.

(ii) We study the Least-Attained-Service (LAS) policy which is viewed as a good substitute for SRPT when the remaining job size is not available [32, 33, 35, 36]. It has been shown that the amount of service a job has received so far is a good indication of its remaining processing time when the jobs size distribution possesses a decreasing failure rate. Well known heavy tail jobs size distribution that is known to accurately describe the actual job size distribution in today's computer systems have a decreasing failure rate. In addition to understanding the delay performance of LAS across job sizes, we analyze the penalty associated with not using the remaining size information directly.

(iii) Instead of analyzing variants of SRPT on a case-by-case basis, we study a class of scheduling policies, SMART [43], that are biased toward smaller job size or remaining size. This class contains many variants of SRPT with different fairness properties. Our analysis shows that all policies in the class have the same tail probability of delay across job size in the many sources large deviation regime.

The results of this dissertation facilitates the understanding of various scheduling policies under the many sources large deviations regime by analyzing the asymptotic delay distribution of previously unexplored policies. The dissertation presents novel methods of queueing analysis based on the 2DQ framework and the cycle concept which have allowed analysis of scheduling policies described in this dissertation and will facilitate understanding of other policies in the future.

1.2 Main Contributions

(1) We study the 2DQ framework [48] along with the cycle based analysis to provide a unifying analytical framework that allows the analysis of various scheduling policies in the many sources large deviations regime previously not understood due to the difficulties brought on by the increased complexity. The proposed methods of analysis demonstrate that most scheduling policies can be viewed in the context of a complex but tractable priority queueing system. This enables the study of a large class of scheduling policies by applying the large deviations results for priority queues.

(2) In this dissertation, the asymptotic delay distribution of scheduling policies that exhibit superior mean delay by favoring small jobs, such as SRPT, PSJF, and LAS, are analyzed in the many sources large deviations regime. In addition, the delay rate functions of all scheduling policies in SMART are shown to be equivalent and is derived. The scheduling policies are compared with each other to understand the benefits and tradeoffs associated with each scheduling policy.

(3) The finite-SRPT policy is proposed to take into account the possible bandwidth constraints at the end receivers. We analyze the rate function of finite-SRPT across job sizes and for varying bandwidth constraint. The results show that finite-SRPT shifts between SRPT and a PS-like policy as the bandwidth constraint is adjusted. Thus, the proposed finite-SRPT can be used as a policy to balance performance and fairness

by simply adjusting the maximum amount a job can be served at any time.

(4) The asymptotic delay distribution in the many sources large deviations regime for Processor–Sharing (in the discrete time framework), a well known fair scheduling policy, is derived.

1.3 Organization

The organization of the dissertation is as follows. The basic system model used throughout this dissertation along with the many sources large deviations regime is explained in Chapter 2. Chapter 2 also briefly introduces many of the scheduling policies mentioned in the dissertation. In Chapter 3, the asymptotic delay tail probabilities across job size are investigated for the SRPT scheduling policy. Then from Chapter 4 to Chapter 6, we address the practical implementation constraints of the SRPT policy. In particular, the delay decay rate of SMART that includes SRPT and its variants is derived in Chapter 4 and the LAS policy which is has been shown to be good substitute for SRPT when the remaining processing time information is not available is studied in Chapter 5. In Chapter 6, we introduce the finite-SRPT scheduling policy that takes into account the possible end user bandwidth constraint and investigates its delay characteristics as the constraint is varied. Then we investigate the well known fair scheduling policy, PS, in the many sources large deviations regime in Chapter 7. Lastly, we conclude in Chapter 8 with a summary of the dissertation and present interesting problems for the future.

Chapter 2

The Basic System Model

Scheduling has found diverse applications in the area ranging from manufacturing, computer systems to flight scheduling and call centers. Depending on the scheduling policy used the performance of the system is greatly affected. Due to its diverse application and its importance, scheduling has been analyzed through a wide range of models and performance metrics. Models range from a simple single server setup where the jobs arrivals and departures are independent, to more complex models where jobs have preferences at which time and at which server it is served. The performance metric under consideration can range from delay, queue size to guaranteed service time and worst case behavior.

Due in part to the wide range in models and metrics, two analytical approaches have received much attention for understanding scheduling. First is the deterministic approach where the worst case behavior of a scheduling policies is of concern. In general, the deterministic approach assumes finite number of job arrivals but makes no assumption on the job size or the arrival sequence. On the other hand, the probabilistic approach is more concerned about the distribution (cdf or pdf) of the metric under consideration. The approach makes probabilistic assumptions on the arrival, service and job size to derive the average, variance and in some cases the distribution of the metric of interest.

In this dissertation, we take the probabilistic approach in analyzing the delay of

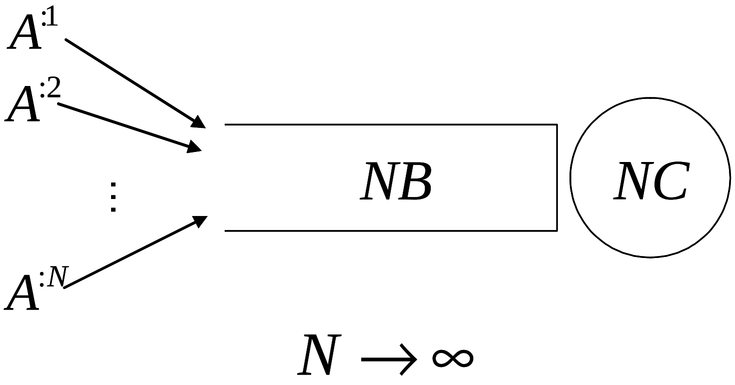


Figure 2.1: Illustration of many sources framework. In the many sources large deviations framework, we consider the probability of large deviations from the mean when the system is accessed by a large number of sources and is served by a server with capacity that increase in proportion.

scheduling policies. We use this approach since we are more interested in what a “typical” job in a system will experience rather than what would happen in the worst possible case. In the rest of the chapter, we describe the system setup used throughout the dissertation in Section 2.1. Then, we go into detail the analytical approach taken and the performance metric under consideration in Section 2.2. Lastly, we give a short introduction to the scheduling policies analyzed and mentioned in this dissertation in Section 2.3.

2.1 The System Setup

We consider a queueing system with a single queue and a single server having stationary and ergodic arrival and service processes, where the arrival and service processes are independent of each other. The system operates in discrete time, i.e. a batch of jobs arrive at the beginning of each time slot and jobs are serviced at the end of each time slot. The queue state is measured immediately after the service and just before the arrivals of the next time slot.

In the many sources regime, the number of arrival processes is scaled along with the capacity of the system and the buffer size as depicted in Figure 2.1. We denote the *i*th arrival process as $A^{:i}$, where $1 \leq i \leq N$. We assume that the possible sizes of the jobs are restricted to bounded multiples of a unit size. Thus, we represent the set of possible job sizes as $\mathcal{M} = \{1, 2, 3, \ldots, M\}$. The assumption that the service distribution is bounded is natural given the numerous recent studies that have observed that file sizes at web servers typically follow a bounded, highly variable distribution size [3, 11]. Formally, for each job size $k \in \mathcal{M}$, we assume N independent, identically distributed processes. We denote $A^N(a, b)$ as the total number of arrivals by all N arrival processes in the time-interval (a, b), where $a \leq b$[1]. For example, $A^N(0, 0)$ signifies the total number of arrivals in time slot 0. Additionally, we define $A_k^N(a, b)$ as the total number of jobs of size k that arrive in the queue during time-interval (a, b). Thus, the volume of size k arrivals is $kA_k^N(a, b)$, and $A^N(a, b) = \sum_{k=1}^{M} A_k^N(a, b)$. We assume independence between arrival processes of different sized jobs, i.e., $A_i^N(a, b)$ is independent of $A_j^N(a, b)$ for $i \neq j$. Note that job arrivals from a single stream of any given size can be correlated across time-slots.

As depicted in Figure 2.1, we assume that the capacity of the server, C, is scaled in proportion to the number of arrival processes, and at most NC data can be service at any time slot. We assume that the server is work-conserving and that the system is stable, i.e. $E\left[\sum_{i=1}^{M} iA_i^N(0, 0)\right] < NC$. Based on these basic setup and assumptions we are interested in the asymptotic (asymptotic in the number of sources) delay distribution in the many sources large deviations regime.

2.2 Many–Sources Large–Deviations Regime

Our goal in this dissertation is to study the tail probability of delay in the many sources regime, i.e., probability of large deviations under the assumption that the number of sources is large. The study of large deviations in a queueing system under an asymptotic regime is

---

[1] The notation (a, b) refers to time slots $\{a, a + 1, \ldots, b\}$.

Notation	*Description*
$\mathcal{M}$	Set of possible job sizes.
M	Largest job size.
N	The scaling factor that represents the number of arrival processes and the multiplicative factor of the capacity and buffer size.
$A^N(a,b)$	Total number of arrivals from all N arrival processes in the interval (a,b).
$A_k^N(a,b)$	Total number of size k arrivals from all N arrival processes in the interval (a,b).
C	The unscaled capacity of the server.
$W^{(N)}(k)$	Actual delay experienced by a size k job in the many sources regime.
$I_W(k)$	Actual delay rate function of size k job
$V^{(N)}(k)$	Virtual delay experienced by a size k job in the many sources regime.
$I_V(k)$	Virtual delay rate function of size k job

Table 2.1: Notations frequently used in the dissertation.

mainly divided into the *large buffer* large deviations and the *many sources* large deviations. The large buffer large deviations is concerned with the tail behavior of delay, $Pr(W(k) > m)$ for large m (where $W(k)$ denotes the delay experience by a size k job). While the large buffer framework has provided many insights about the behavior of the tail of $W(k)$, this type of analysis leaves several questions unanswered. Most importantly, the large buffer framework only studies the *extreme tail* behavior of $W(k)$, i.e. $Pr(W(k) > m)$ as $m \to \infty$, rather than the full *distribution* of $W(k)$.

In this dissertation, we take a different approach and analyze $W(k)$ in the *many sources* large deviations regime. To reflect the large number of sources considered, we denote $W^{(N)}(k)$ as the delay experienced by a size k job in the many sources regime. While the large buffer large deviations studies the overflow probability of a single queue and single arrival process as the buffer size goes to infinity, the many sources large deviations scales the service capacity with the number of arrivals (see Figure 2.1). Thus, the many sources regime has the advantage that it allows us to study $Pr(W^{(N)}(k) > m)$ for all m, rather than requiring that m be large. That is, it allows us to obtain information about the distribution of $W^{(N)}(k)$, rather than just the tail. Another advantage of the many sources regime is that scaling the number of sources (arrivals) seems appropriate for studying scheduling in web applications given the high level of statistical multiplexing among multiple user requests, while scaling the delay threshold m seems less appropriate given the finiteness of delay in web applications.

Formally, $W^{(N)}(k)$, is the delay experienced by the last job with size k in an arrival burst to a stationary system. The tail probability of delay, $\Pr(W^{(N)}(k) > m)$, is the probability that the last job of size k in the burst arriving at time slot l does not leave the system by the end of time slot $l + m$. It has been shown that, in the large deviation framework, the tail probability of delay of various scheduling policies such as FIFO, Generalized Processor Sharing (GPS), and Priority Queueing decays as

$$\Pr(W^{(N)}(k) \geq m) = g^N(k,m)e^{-NI_W(k,m)},$$

under general conditions, where $\lim_{N\to\infty} -\frac{1}{N}\log g^N(k,m) = 0$. In other words, the most dominant trend of the tail probability is the exponential decay $I_W(k,m)$, which is appropriately called the *decay rate*. Thus, the decay rate of delay is defined as

$$I_W(k,m) = \lim_{N\to\infty} -\frac{1}{N}\log \Pr(W^{(N)}(k) > m). \tag{2.1}$$

In this dissertation, we show that such decay rates for the scheduling policies under consideration do exist and derive their precise form. Note that the delay distribution for a job of size k depends on the capacity C, the threshold value m, the job size k, and the arrival processes $A_k^N(a,b)\ \forall k \in \mathcal{M}$. In particular, the contribution of the arrival process to the delay decay rate of scheduling policies is expressed through their own decay rate, which has been well analyzed in the literature, i.e.,

$$I_{A_k}^{(a,b)}(x) = \lim_{N\to\infty} -\frac{1}{N}\log \Pr(A_k^N(a,b) > Nx). \tag{2.2}$$

Though our goal is to study the decay rate of delay, the direct derivation is difficult, so we first consider the distribution of the *virtual delay*. The *virtual delay*, $V^{(N)}(k)$, is the delay seen by a fictitious (virtual) job that arrives at Q_k, queue for size k jobs, at the end of an arrival burst at $t = 0$ (given that the system started at $t = -\infty$). The event $\left\{V^{(N)}(k) > m\right\}$ corresponds to a fictitious job arriving at the end of an arrival burst during time slot 0 and not departing the system until the mth time slot. Note that this setup ensures that the system is stationary at the arrival of the virtual job. To avoid confusion, we will refer to the delay $W^{(N)}(k)$ as the *actual delay* in order to distinguish it from the *virtual delay* $V^{(N)}(k)$. Observe that the virtual delay is different from actual delay: for example, even when there is no arrival the virtual delay can be measured, whereas the actual delay is not defined. The decay rate of the virtual delay is defined as

$$I_V(k,m) \quad = \quad \lim_{N\to\infty} -\frac{1}{N} \log \Pr(V^{(N)}(k) > m). \tag{2.3}$$

In the our proofs, it will be necessary to have a more general definition of the decay rate than we have defined so far. Thus, for any sequence of rare events $\mathbf{H}^N$, we define

$$\mathbf{I}(\mathbf{H}) \quad = \quad \lim_{N\to\infty} -\frac{1}{N} \log \Pr\left(\mathbf{H}^N\right),$$

as the decay rate of a general sequence of events $\mathbf{H}^N$ whose probability becomes increasingly small as the system scales.

2.3 Brief Overview of Scheduling Policies

Since the main focus of this dissertation is to understand the effects different scheduling policies have on the delay distribution, a short overview of the various scheduling policies are given in Table 2.2. By no means complete, we provide Table 2.2 as an reference point for abbreviated names of the scheduling policies. The terms original size, remaining size, and attained service of a job is used throughout this dissertation to signify the following information of a job that are used by the scheduling policy. The original size of a job is the

$SchedulingPolicy$	$Description$
FCFS	First–Come–First–Serve serves jobs that have arrives first. Also known as First–In–First–Out (FIFO)
PRI	Priority queueing system with multiple queues where jobs of smaller original size reside in queue with higher priority.
SJF	Shortest–Job–First serves jobs of smaller original size first.
PSJF	Preemptive–Shortest–Job–First preemptively serves jobs of smaller original size first.
SRPT	Shortest–Remaining–Processing–Time preemptively serves jobs with smaller remaining size first.
LAS	Least–Attained–Serves preemptively serves jobs with least attained service first.
PS	Processor–Sharing serves all jobs the same rate.
RS	RS preemptively serves jobs with the least product of remaining processing time and original size

Table 2.2: Overview of scheduling policies.

processing time required for a job to leave the system when it arrives to the system. The remaining size of a job is the remaining processing time left for the job to leave the system. The attained service of a job is the amount of processing time that has been served so far.

Note that the table contains policies that are *non-preemptive* and ones that are *preemptive*, where the scheduling policies are non–preemptive unless stated otherwise. Preemptive policies allow a job to be suspended during service when a higher priority job arrives to the system, while a non-preemptive policy guarantees that a job will be fully served when it begins service.

We make the following assumptions on the scheduling policies under consideration in this dissertation. All scheduling policies are *work-conserving* which means that as long as there are jobs in the system the server serves jobs with its full capacity. In addition, we assume that there is no cost or penalty in being preempted and that a job resumes service starting from where it has left off.

Chapter 3

SRPT

3.1 Background and Related Work

With web service becoming increasingly popular, today's web servers handle loads that could range from hundreds to thousands of simultaneous connections. These jobs are served by a web server by means of a scheduler, which "prioritizes" the requests. Many different scheduling policies have been proposed in the literature and implemented in practice. It is well known that Shortest-Remaining-Processing-Time (SRPT), which gives higher priority to jobs with a smaller remaining processing time[1], exhibits the minimum mean delay among all policies. Despite this advantage, most of the existing schedulers opt for simpler policies such as First-Come-First-Serve (FCFS), or fairness oriented scheduling policies where the resources are equally shared among all connections. The main reason behind the lack of attention to SRPT is that, it has been believed that in the process of optimizing the mean delay, fairness among jobs of different sizes might suffer, i.e., "starvation" of larger jobs [39]. More specifically, it is believed that larger file requests will "starve" under the SRPT scheduling policy [39, 40]. Intuitively this seems obvious – by giving priority to smaller file requests, the delay experienced by larger file requests will increase, thus leading to

---

[1] More specifically, SRPT serves jobs with the smallest remaining processing time first and in the case of a tie the jobs are served in FCFS manner.

unfairness.

However, recent studies have shown that this intuition is not necessarily correct. In [4], the authors show that under a heavy-tailed arrival distribution, the unfairness of SRPT compared to a Processor-Sharing (PS) scheduling policy is quite small. Using an M/G/1 queuing model and a heavy-tailed arrival distribution, the authors have shown that regardless of the load, at most only 1% of jobs have a larger expected delay under SRPT than under PS. Further, the authors provide an upper bound on how much worse SRPT can perform compared to PS in terms of expected delay. Motivated by these results, the authors have implemented an SRPT scheduler for web servers [18], which has been shown to dramatically reduce delay.

Thus, the SRPT scheduling policy is a promising alternative to the prevalent scheduling policies. However, there are realistic concerns that the fairness problem in SRPT may deteriorate when the system is accessed by a large volume of traffic. In addition, the mean delay metric is not enough to understand the characteristics of a scheduling policy, especially when one is interested in the QoS of the system. To realize its full potential, the analysis of the delay distribution for specific sized jobs in a large scale system is necessary in order to characterize the behavior of this policy. More specifically, we will be looking at the probabilities of the delay exceeding some threshold (tail probability), when the web server is accessed by a large number of sources, using a large deviations formulation.

Related works include [7] where SRPT is analyzed in the large buffer regime, and [19] where it has been shown that various scheduling policies including SRPT and PS have the same *expected* slowdown[2] for asymptotically large sized jobs. However, there is no work in the many sources large deviation regime which is geared to understand web servers or routers in large scale systems. Our work focuses on the many sources regime, and compares the SRPT scheduler with a FCFS scheduler.

The main intuition behind the analysis of SRPT is the utilization of **virtual queues**

---

[2] Slowdown is defined as delay divided by the job size.

and the **large scale limit**.

(1) **Virtual Queues**: The scheduling policies in question possess only a single physical queue. Thus it is difficult to track the state change that a job goes through in this single queue. However, scheduling policies distinguish jobs by their state information, e.g., time of arrival, original size, remaining size, and attained service, to decides which job to service next. In essence, all scheduling policies assign priority to jobs based on their state and serve them in a specific order. We construct a system of virtual queues which reflects this priority scheme of a specific scheduling policy. This construction of virtual queues for analysis purpose enables the understanding of the state changes of the system at all times. The scheduling system is re-interpreted through this system of virtual queues to derive the delay distribution in the many sources large deviation framework.

(2) **Large Scale Limit**: We observe that in the many sources large deviations regime only the main characteristics of the scheduling policy remain as the scale increases. In other words, the delay distribution of scheduling policies in a large scale computer network can be described by its first order behavior, and all other higher order behaviors can be ignored.

3.2 Main Contributions and Intuition

In this section, we show that the rate function of delay for SRPT is equivalent to that of a priority queue that assigns higher priority to jobs with smaller original size. Let us denote PRI as a priority queueing system described below. Formally, SRPT and PRI are defined as follows.

SRPT: In any time slot, jobs that have the smallest remaining processing time are served first. When there are multiple jobs that have the same smallest remaining processing time, the job that arrived earlier to the queue is selected to be served. The job in service receives service until either it is fully served or new jobs with even smaller remaining processing time arrive to the system at the next time slot. In the latter case, the job becomes preempted and waits until all jobs that have higher priority (smaller remaining processing time) leave the system.

PRI: PRI is a non-preemptive priority queueing system, where jobs of different sizes are queued separately, and queues that are assigned to smaller sized jobs have higher priority, while the queues assigned to larger sized jobs have lower priority. For example, the queue that corresponds to size l jobs (henceforth referred to as Q_l) has priority l, where Q_1 has the highest priority, and Q_M has the lowest priority. FCFS rule is observed among jobs in the same queues, i.e., jobs of the same original size. PRI is different from SRPT in that the job in service does not get preempted when new jobs arrive to the system in the next time slot. Thus a job in service continues to receive service until it leaves the system.

We prove that the rate function of the SRPT scheduler is bounded by the rate functions of the priority queueing schedulers, and use results on priority queues from [38] to complete the proof. First, we make the following two assumptions on the arrival processes.

Assumption 3.2.1. [3] *Fix any* $\epsilon > 0$*, such that* $E[A^N(0,0)] < N(C-\epsilon)$*, and consider the event*

$$\mathbf{H}_T^\epsilon = [\sum_{i=1}^{k-1} A_i^N(-T,m) + A_k^N(-T,0) > N(C-\epsilon)(T+m+1)].$$

We define $T^* = \arg\inf_{T\geq 0} \mathbf{I}(\mathbf{H}_T^\epsilon)$*. The existence of* T^* *results from the stability condition and we assume that it is unique. Furthermore, we assume* $\mathbf{I}(\mathbf{H}_T^\epsilon)$ *satisfies the following*

---

[3] This assumption can be shown to be satisfied by sources with bounded rates [24], i.e., arrival processes for which the total number of arrivals per slot is bounded.

condition

$$\liminf_{T\to\infty} \frac{\mathbf{I}(\mathbf{H}_T^{\epsilon})}{\log T} = \omega > 0. \tag{3.1}$$

More specifically, $\mathbf{H}_T^{\epsilon}$ is the following event: The event that the sum of all arrivals with job size $\leq k$ during the time interval $(-T, 0)$ and the arrivals due to jobs with size $\leq (k-1)$ over the interval $(1, m)$, exceeds the service capacity of $N(C-\epsilon)$. Large deviations theory roughly states that the probability of occurrence of a rare event is dominated (in a large-scale system) by a single "critical" event. It is shown in Theorem 3.2.1 that the event $\mathbf{H}_T^{\epsilon}$ is the "critical" event of $\{V^{(N)}(k) > m\}$. Equation (3.1) can be shown to be satisfied by sources with bounded rates [24], i.e., arrival processes for which the total number of arrivals per slot is bounded.

Assumption 3.2.2. *("Burstiness" condition) Define*
$\vec{A}_i = (\ldots, A_i^N(-T, 0)\ldots, A_i^N(-1, 0), A_i^N(0, 0))$. *Then* $\vec{A}_i$ *satisfies*

$$(\vec{A}|A_i^N(0,0) = 0) \leq_{st} (\vec{A}|A_i^N(0,0) > 0), \tag{3.2}$$

where $\leq_{st}$ *denotes stochastic ordering.*

This "'burstiness" condition is used in essence to bridge the gap between the derived virtual delay rate function of SRPT and the actual delay rate function. We show that from the results in [38] and Assumption 3.2.2, the virtual and actual delay asymptotes are the same. The above assumption heuristically corresponds to a "burstiness" condition on the arrival process, see [38]. Many types of sources satisfy this condition, including any ON-OFF Markov sources.

Based on the two assumptions described above, the delay rate function of SRPT size k (for any $k \in \{1, \ldots, M\}$) jobs has been derived as in Theorem 3.2.1 [46] under Assumption 3.2.1 and Assumption 3.2.2 .

Theorem 3.2.1. *Fix any $\epsilon > 0$. Then, for any $k \in \{1, \ldots, M\}$ we have*

$$I^{(k)}_{V_{C-\epsilon}}(m) \leq I^{(k)}_{\overline{V}}(m) \leq I^{(k)}_{V_{C+\epsilon}}(m),$$

where we denote by $I^{(k)}_{\overline{V}}(m)$, the delay rate function of size k jobs under SRPT with total capacity NC. Similarly, $I^{(k)}_{V_\mu}(m)$ denotes the delay rate function of size k jobs under SJF, and with total capacity $N\mu$.

Proof. Intuition (technical proof in Appendix A.1): We make the observation that the operation of SRPT is equivalent to that of PRI described above but with switching of partially served jobs during time slots. The SRPT scheduler grants jobs with less remaining service time higher priority compared to jobs with larger remaining service time. Jobs that have not been completely serviced (partially served) in a time-slot will receive higher priority in the next time-slot depending on their remaining service time. Thus, the SRPT scheduler can be modeled as a priority queueing system, where the queues are not assigned by the job size but the residual job size (remaining service time of a job). Another way to interpret SRPT is to look at SRPT as being equivalent to the priority queueing system based on job sizes, but with partially served jobs changing priority levels at the end of a time-slot (moving to a corresponding higher priority queue). Note that an important difference between the original priority queueing system and the SRPT scheduler is that, in the priority queueing scheduler a partially served job continues to remain in the same queue that it was originally in, and its priority level *does not change* in the next time-slots.

Next, we show that the number of preempted jobs in SRPT is very small compared to jobs that do not get preempted. The intuition for this is due to the observation that at most only a single job can be preempted in any time slot, whereas a large number of new jobs arrive in each time slot. As a result, the number of partially served jobs that change priority levels is negligible, i.e., the number of jobs that are served with respect to remaining processing time different from their original size is negligible. Thus, we can conclude that the delay rate function of SRPT is equivalent to that of PRI in the many source large deviations regime. This result on the virtual delay can be extended to the case of actual delay under Assumption 3.2.2 as shown in [46]. □

Using the delay rate function result of SRPT, we investigate the performance and fairness of SRPT in the large scale regime by comparing the delay rate functions of SRPT and FCFS (a scheduling policy that does not distinguish jobs of different size). A FCFS queueing system consists of a single server and a single queue, and jobs are processed in the order they arrived. The comparison of the delay rate function is derived in Theorem 3.2.2 [46] under Assumption 3.2.3.

Assumption 3.2.3. [4] *We assume the following on the marginal probabilities of arrival,* $A_k(0,0)$*, for all* $k \in \mathcal{M}$

$$A_k(0,0) = \begin{cases} 0 & \textit{with probability } p(k) \\ k & \textit{with probability } q(k) = 1\text{-}p(k), \end{cases}$$

where $q(k)$ *satisfies: for any* $\eta < 1$ *there exists constants* $A_\eta \geq 1$*, and* $K_\eta \geq 1$ *such that for all* $k \geq K_\eta$

$$q(k)e^{k^\eta} \geq A_\eta.$$

Assumption 3.2.3 states that the marginal probability of the arrival for large sized jobs *does not* decay exponentially in job size, i.e., it decays more slowly than exponential. Note that this assumption does not place restrictions on the time correlations.

Arrival processes that have this property include arrival processes with a truncated heavy tailed job size distribution[5]. We note that, Assumption 3.2.3 holds not only for trun-

---

[4] Assumption 3.2.3 holds for all arrival processes which *do not have an exponential tail*. As special cases, heavy tailed arrivals as well as truncated heavy tailed distribution possess this property

[5] It has been observed that job sizes on web servers typically follow a heavy-tailed distribution with an appropriate max job size (i.e. truncation). This property is quite common in web workloads; the heavy tailed property is seen in the distribution of job size requested by clients, the length of network connections, and jobs stored on servers [3, 11, 12]. By heavy tailed distribution, we mean that the tail of the distribution function de-

cated heavy-tailed arrivals, but for all arrival processes which *do not have an exponential tail* (heavy-tailed arrivals are just a special case for which the assumption holds).

Under Assumption 3.2.3 on the marginal probabilities of the arrival process, the upper bound on the difference of the delay rate function for size k jobs in SRPT and FIFO is derived. We denote $I_{\hat{V}}(m)$ as the delay decay rate of jobs in FIFO. We comment that the delay rate function for FIFO is invariant with respect to job size. In other words, the virtual delay seen by size 1 jobs is the same as any other.

Theorem 3.2.2. *An upper bound of the difference between the rate function of size k jobs for SRPT and FCFS is as follows. For any* $0 < \gamma < 1$, *there exists a* $\hat{K}(m)$ *such that*

$$|I_{\overline{V}}^{(k)}(m) - I_{\hat{V}}(m)| = O(\frac{1}{k^{\gamma}}) \quad \forall k \geq \hat{K}(m).$$

Proof. The technical proof is provided in Appendix A.2. □

The above result implies that *the difference of the delay rate functions for size k jobs under SRPT and FIFO decays at least as fast as* $O(1/k^{\gamma})$ *for k large*. Thus, even though the job size is increasing, the delay performance of SRPT approaches that of FIFO in the many sources regime. This result shows that the unfairness of SRPT (compared to FIFO) becomes increasingly small for increasingly large jobs. On the other hand, for *size 1 jobs*, the difference of the delay rate functions remains constant. This is due to the fact that the delay distributions under SRPT is *invariant with k*, i.e., larger jobs do not influence the delay of smaller jobs under SRPT scheduling. However for FIFO, the delay distributions for even *size 1 jobs* decays as $O(\frac{1}{M^{\gamma}})$ for M large. Thus, these results indicate that SRPT is a good policy to implement for web-servers, where empirical evidence suggests a large variability in job sizes [2], since the unfairness of larger jobs is small compared to the benefits gained for the smaller jobs.

---

cays with a power law with exponent less than 2. That is, if a random variable X has a heavy-tailed distribution, then $\Pr(X > x) \sim x^{-\alpha}$ for $0 < \alpha < 2$. We can see that for heavy-tailed arrival processes, the arrival probability of large sized jobs decays only polynomially (slower that exponential, i.e., Assumption 3.2.3 is satisfied for $\eta > 0$).

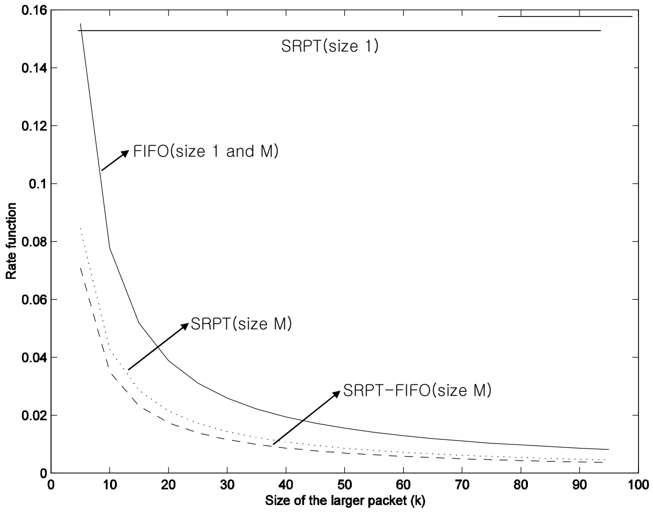


Figure 3.1: Illustration of the delay decay rate of size 1 and M jobs for SRPT compared with FIFO. The decay rate of size 1 jobs for SRPT is actually infinite since the setup ensures that all size 1 jobs are served as soon as they arrive. It is depicted as a horizontal line for viewing purpose. Note that the decay rate of size M jobs for SRPT approaches that of FIFO as the M increases while the difference for size 1 jobs remains large (actually grows for large M).

3.3 Numerical Analysis

In this section, we compare the delay rate functions of SRPT and FIFO numerically. We consider a system where the arrival process is assumed to be composed of two independent ON-OFF processes which are one of two types: size 1 jobs arrive with probability p, and size M jobs arrive with probability p/M. The numerical values of the delay rate functions are calculated for $C = 0.9, p = 0.4, m = 2$ using the closed form expressions derived in this dissertation. The result comparing size 1 and size M jobs is depicted in Figure 3.1. Figure 3.1 shows that the difference of the delay rate function of size M jobs between SRPT and FIFO becomes smaller as M increases. We conclude that even in the extreme scenario, the effect of increased delay for larger jobs in SRPT compared to FIFO becomes smaller in the asymptotic sense. In comparison, the rate function for size 1 jobs of SRPT is superior of that of FIFO by at least factor of 100. As shown in Figure 3.1 and Theorem 3.2.2, the difference in the delay rate function for larger jobs between SRPT and FIFO indicates that the delay rate function of SRPT approaches that of FIFO for increasingly large jobs while the delay performance of SRPT for smaller jobs remains much better than FIFO. It has been shown that web server requests exhibit heavy-tailed arrival distribution [2]. The two classes traffic model that we studied approximates such a heavy traffic behavior for large M. Thus, the results indicate that SRPT is a promising scheduling policy which can be readily employed in web-servers.

Chapter 4

SMART

4.1 Background and Related Work

In the previous chapter, we considered SRPT. In this chapter, we consider a class of SRPT-like policies called SMART [43], which performs well for a combination of metrics such as mean delay, mean slowdown[1], and fairness [17, 28, 33, 34]. SMART contains SRPT and many of its variants whereby the analysis of SMART allows the understanding of many of the practical implementations of SRPT.

SMART was introduced by Wierman and Harchol-Balter in [43] and is formally defined as follows. Denote jobs using a, b, and c where job a has original size s_a and remaining size r_a. SMART is defined to be the class of scheduling policies that obey the following properties.

(1) **Bias Property:** *If* $r_b > s_a$*, then job a has priority over job b.*

(2) **Consistency Property:** *If job a ever receives service while job b is in the system, thereafter job a has priority over job b.*

(3) **Transitivity Property:** *If an arriving job b preempts job c; thereafter, until job c*

---

[1] Slowdown of a job is the delay divided by the job size.

receives service, every arrival, a, with size $s_a < s_b$ *is given priority over job c.*

The Bias Property guarantees that scheduling policies in SMART favor “small” jobs by guaranteeing that the job receiving service has a smaller remaining size than the original size of all jobs in the system. This ensures that the server will not service a new arrival with greater size than an existing one. As observed in [43], this Bias Property is the key property that allows SMART do the “smart” thing.

The Consistency and Transitivity Properties essentially ensure the “coherency” of the priority scheme dictated by the Bias Property. In particular, the two properties are concerned on the priority scheme after jobs have received partial service. The Consistency Property effectively prevents time-sharing by enforcing the rule that once a job receives service other jobs already in the system do not receive service until the former job leaves the system. In other words, if job a is given priority over job b, then job b will never served before job a. The Transitivity Property ensures that SMART do not second guess the decision they have made. For example, if it is decided that job a is smaller (have higher priority) than job b, new arrivals that are smaller than job a are considered smaller than job b.

As shown in [43], SMART contains many important “smart” policies such as SRPT, PSJF, and a wide array of hybrid policies with more complicated prioritization schemes. In particular, an interesting scheduling policy called RS policy, which assigns higher priority to jobs with smaller product of its remaining size and its original size, is included in SMART. The RS policy is interesting in that the policy outperforms SRPT if when we consider weighted mean delay measures such as the mean slowdown. SMART actually includes many generalization of these policies. In particular, it has been shown in [43] that scheduling policies that grants priority based on a fixed priority function $p(s, r)$ such that for $s_1 \leq s_2$ and $r_1 < r_2$, $p(s_1, r_1) < p(s_2, r_2)$ are contained in SMART. An example of such a policy is a policy that has $p(s, r) = s^i r^j$ for all $i \geq 0$ and $j > 0$.

SMART is important due to the fact that the class includes a wide range of policies that perform well for a combination of metrics such as mean delay, mean slowdown, and delay tail probability. For these combinations of metrics the optimal policy is not SRPT, rather the optimal scheduling policy depends on the specific job size distribution. The importance of SMART lies in the fact that many of the optimal scheduling policies with respect to mixed metrics are included in it, since no single policy in SMART is optimal across applications. In addition, SMART contains time-varying policies so it contains policies that can optimize performance online in the face of time varying performance metric, system-state information, and randomization.

Despite its breadth, many "smart" policies are excluded from SMART, e.g. LAS, and Shortest-Job-First (SJF). However, the exclusion of such policies is due to the goal in defining a class of policies that are near optimal in terms of mean delay across all service distributions and all loads. For example, SJF exhibits arbitrarily large mean delay when the second moment of the service distribution is large, and LAS is the worst policy when the job size distribution is of increasing failure rate. One motivation for working with SMART is to illustrate the wide range of policies that behave like SRPT with respect to mean delay.

Recent studies of policies that prioritize small job size, such as SRPT and SMART, have focused on the delay experienced by a job of size k, $W(k)$. The interest in $W(k)$ is spurred by the desire to understand how the prioritization of small job sizes affects the behavior of $W(k)$ across jobs sizes. In particular, there are worries about the delay experienced by large k, which the scheduling policies are biased against. Much attention has been given to understanding $E[W(k)]$, the mean delay experienced by a job of size k, across k [1, 4, 32, 33, 41]; however far less is understood about the *distribution* of $W(k)$ in large scale systems.

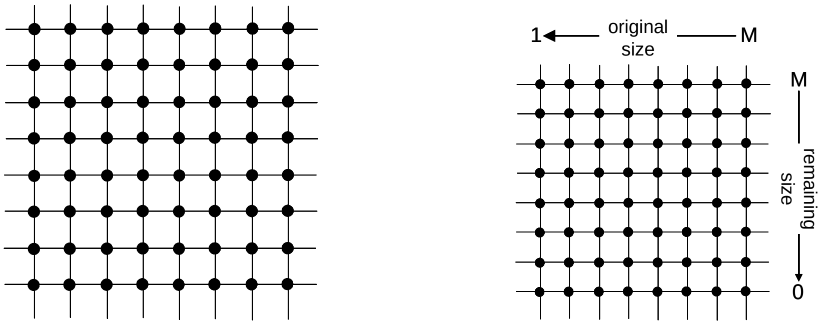
The difficulty in direct analysis of the distribution of $W(k)$ has led researchers to study asymptotic scalings of the distribution. The many sources large deviations is one such asymptotic approximation of the distribution suitable for understanding large scale systems. In this chapter, we analyze the delay distribution of SMART that contains scheduling policies that biased toward jobs of small original size and/or remaining processing time.

4.2 Two Dimensional Queueing Framework

In order to analyze SMART in the many sources regime, we study the *two dimensional queueing* (2DQ) framework [48]. The basic 2DQ framework is a collection of virtual queues as depicted in Figure 4.1(a), which in essence allows us to look at a scheduling policy as a complex but tractable time varying priority queueing system. Note that the idea of virtual queues used in Chapter 3 is extended to an array of virtual queues in the 2DQ framework. The 2DQ framework is based on the idea of suitable and coherent arrangement of multiple *virtual queues* to represent the state of the system.

The state of the system is adequately represented by a finite number of *virtual queues*, i.e., the virtual queues are defined and arranged in such a way that it represents the state of the system at all times. The collection of virtual queues contain jobs of certain state so that the collection completely describes the state of the system at all times, where an individual virtual queue contains jobs of a certain state. The fact that the state of the system can be represented by the two dimensional collection of virtual queues (2DQ) is made possible by the fact that the number of all relevant job states is *discrete*. This is due to the following two assumptions made in Chapter 2. First, the job sizes are restricted to multiples of a unit size. Second, the server serves jobs in discrete amounts. More importantly, the 2DQ framework allows the delay analysis of SMART by providing a coherent and tractable portrayal of the system state changes as the jobs receive service based on the following two concepts: *finiteness* and *ordering*.

First, by *finiteness*, the number of virtual queues in 2DQ required to fully represent the state of the system is finite, which makes the analysis much simpler. We have assumed that the server services the jobs in *discrete* amounts and that the set of possible job sizes is $\mathcal{M} = \{1, 2, 3, \ldots., M\}$. The important consequence is that at any time slot, the distribution of all relevant states of any job in SMART (original size and remaining size) is *discrete* and



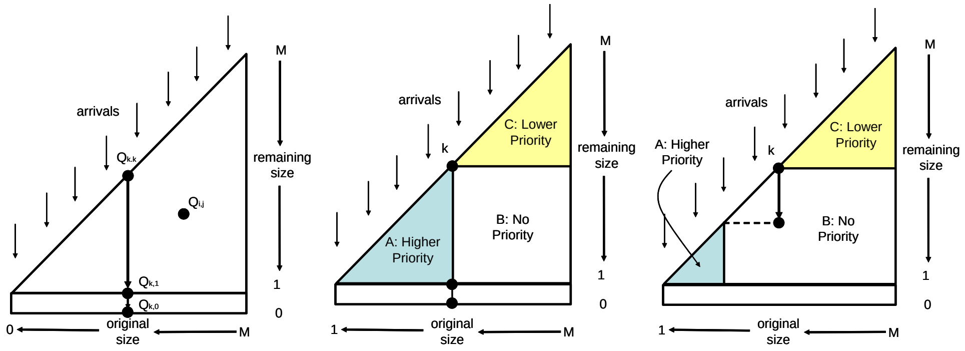
(a) The basic Two Dimensional Queueing (2DQ) Framework.

(b) Two Dimensional Queue for SMART.

Figure 4.1: Illustration of the Two Dimensional Queueing (2DQ) Framework. The left figure depicts the basic form of the 2DQ framework which is basically a collection of virtual queues arranged in a grid to reflect the state of the system. The right figure depicts the 2DQ representation for SMART where the X-axis is the original size and the Y-axis is the remaining processing time of a job.

finite. This results in a more tractable description of the queue state.

The second important concept of the 2DQ framework is *ordering*. Many scheduling policies specify a scheme of ordering (prioritization), in which the server considers some jobs more important than others and serves those first. For example, FCFS orders jobs by their time of arrival, SRPT by the remaining size (i.e., remaining service requirement), and LAS by their age (i.e., attained service). We make the observation that all jobs in the system must be totally ordered by the specific scheduling policy. In other words, the concept of ordering should be taken one step further by assigning a further ordering schemes beyond the previously existing one to specify a total ordering or the lack of it. *The two dimensional queueing framework is a collection of virtual queues that portray this ordering of a scheduling policy in such a manner that the two dimensional queue separates high, low and no priority set of virtual queues in a coherent and consistent manner.* For example, SRPT does not specify any rule for jobs with the same remaining size. To make the operation of the policy more tractable, we introduce an additional ordering to be used to determine which of these jobs should be served first. Based on this observation, the 2DQ



(a) (b) (c)

Figure 4.2: Illustrations of the two dimensional queueing framework for SMART. Note that since a job cannot have remaining size larger than its original size only the lower right triangle of Figure 4.1 is of importance. The progression of a job between queues while in the system is illustrated in (a). The priority structure for an incoming job is shown in (b) and for a partially served job is shown in (c).

framework takes the concept of ordering inherent in the given scheduling policy one step further by assigning a secondary ordering scheme to the inherently existing one. In other words, jobs are serviced in the order specified by the scheduling policy, but when there are multiple jobs with the same priority, the secondary ordering scheme is used to select the next job to serve. For example, let us consider the ordering of “smaller job size first” as the secondary ordering for FCFS. In this case, when multiple jobs arrive to the system at the same time slot, smaller jobs are served before larger jobs. The importance of the secondary ordering is that it further constrains the policy, thus making the analysis more tractable. In the case of SMART, the policy itself specify all the necessary ordering required to embed it into a two dimensional queueing framework and only the judicious ordering of the virtual queue is required. The two dimensional queueing framework for SMART is depicted in Figure 4.1(b).

4.3 Intuition

Note that SMART falls into the 2DQ framework very nicely as shown in Figure 4.2. A new job arrives to the queues on the upper most strip, where the original size and the remaining size are equal. The job then progresses through the system by moving downward until the remaining size becomes 0. We denote by $Q_{i,j}$, $i \geq j$, the virtual queue that contains all the jobs that have original size i and remaining size j. For example, a job of size k arrives at $Q_{k,k}$, and then moves through $Q_{k,k-1}$, $Q_{k,k-2}$, ..., $Q_{k,1}, Q_{k,0}$ as a unit service is received. It is important to note that the properties of SMART dictates the areas of high, low, and no priority with respect to the job in question as depicted in Figure 4.2.

Let us now consider the behavior of a size k job under a policy of SMART. Upon arrival, the job resides in $Q_{k,k}$. By the Bias Property (LD), the following queues have higher and lower priority compared to $Q_{k,k}$:

$$\text{Higher Priority: } \bigcup_{i=1}^{k} \bigcup_{j=1}^{k-1} Q_{i,j}, \quad \text{Lower Priority: } \bigcup_{i=k+1}^{M} \bigcup_{j=k}^{M} Q_{i,j}. \tag{4.1}$$

This is illustrated in Figure 4.2(b). We denote the group of higher priority queues as area A and lower priority queues as area C. Note that, there is a third area where the queues do not have any fixed priority order compared to $Q_{k,k}$. Jobs in this area, B, can be serviced before or after the size k job. Due to the priority scheme, area A must be empty for the job in $Q_{k,k}$ to receive one unit of service. As depicted in Figure 4.2(c), when a job in $Q_{k,k}$ receives service it moves down the vertical line in the two dimensional queue and the respective priority areas change according to the position of the queue in which the job resides. For a partially serviced size k job to receive service, all queues in the corresponding area A, as depicted in Figure 4.2(c), must be empty. In both cases of an incoming size k job and partially serviced size k job, the relative priority of queues in area B is unknown. However, the volume of jobs in area B can be sufficiently bounded to provide tight upper and lower bounds to derive the probability of delay in the many sources large deviations regime of SMART.

4.4 Main Contributions

4.4.1 SMART for Large Deviations

SMART identifies a group of policies that are similar to SRPT and PSJF (policies biased toward smaller jobs) with respect to the *mean delay* metric in the traditional model when there is only a single flow and the capacity does not scale with the flow. A natural question to ask in the context of the many sources large deviations regime is the following. Is there a class of scheduling policies that exhibit similar or identical characteristics with respect to the *delay distribution in the many sources large deviations regime*? In this dissertation, we observe that indeed such a class exists and we denote it as SMART-LD, i.e., SMART for Large-Deviations. We define SMART-LD as follows.

Definition 4.4.1. *A scheduling policy satisfying the following property belongs to **SMART-LD**.*

(i) ***Bias Property (LD):*** *If $r_b \geq s_a$, then job a has priority over job b.*

Note that the Bias Property changed from a strict inequality (>) to an equality (≥). This change in the Bias Property guarantees the total ordering which is one of the key intuition behind the 2DQ framework. More specifically, the change guarantees that, among jobs with equal original size, jobs with smaller remaining size are given higher priority, and among jobs with equal remaining size, jobs with smaller original size are assigned higher priority.

Further, note that the Consistency Property and the Transitivity Property of the original SMART are excluded in SMART-LD. We show that even without the two properties that enforce “coherency”, their contribution to the delay distribution becomes insignificant when the scaling constant, N, scales to infinity. The exclusion of the Consistency Property and the Transitivity Property in SMART-LD implies that SMART-LD contains all scheduling policies in SMART *and* more complicated policies for which the “coherency” of the priority scheme is lost. In particular, this means that scheduling policies that time share the capacity and/or decide to change priority after a job has been served are included in SMART-LD. This is surprising in that SMART-LD includes SMART like policies that have low level of coherency, where the priority between jobs can change after they have been served while possessing the same delay characteristics in the many sources large deviations regime.

The importance of SMART-LD lies in the fact that (i) it inherits all the aspects of SMART, i.e., inclusion of all scheduling policies that are close to optimal in terms of the mean delay, and (ii) it contains all policies that exhibit identical delay tail probability characteristics in the many sources large deviations regime. The latter is important because (i) SMART-LD attempts to identify all scheduling policies that behave the same in terms of delay in the many sources regime to the mean delay optimal SRPT, and (ii) by deriving the delay distribution of SMART-LD, the delay characteristics for many “smart” policies when accessed by a large number of sources can be understood.

It is important to note that the proposed SMART-LD essentially makes the interior of areas A and C of the virtual queues also “no priority”. The main intuition is that since the number of jobs in the interior of the 2DQ representation (see Figure 4.2) are negligible in the many sources large deviations regime the derivation of the delay rate function of SMART-LD requires only a slight adjustment to that of SMART which is also the result of this dissertation.

4.4.2 Analytical Result

The main contributions regarding SMART-LD is the derivation of the delay rate function. We make two assumptions for the analysis, one of which is the following.

Assumption 4.4.1. *We assume that the rate function corresponding to the arrival* $A = \sum_{i=1}^{M} A_i$ *satisfies*

$$I_A^{(-T,l)}\left(C(T + l + 1) - v\right) < I_A^{(-T,0)}\left(C(T + 1)\right), \tag{4.2}$$

for $v \in [v^* - \delta, v^* + \delta]$*,* $v^* > 0$ *and* $\delta > 0$ *sufficiently small.*

Assumption 4.4.1 is equivalent to the decay rate being additive. Intuitively, a decay rate with the property of additive functionals implies that the occurrence of a rare event in the large deviation framework happens in a straight line. This assumption has been used extensively in the large deviation literature [5, 14, 27]. Further, arrival processes that satisfy Assumption 4.4.1 include many common processes such as all stationary and Markov dependent processes. Additionally, if the arrival process is of Levy type then the decay rate of the arrival satisfies Equation (4.2).

The second assumption is similar to Assumption 3.2.2 but for an event that corresponds to the higher priority jobs in SMART. For brevity, we will not go into detail on the specifics of the assumption.

We prove that the delay rate function experienced by size k jobs is the same under all policies in SMART and derive the delay rate function of SMART-LD as follows [48].

Theorem 4.4.1. *Let* $\epsilon > 0$*. For any* $k \in \mathcal{M}$*, the decay rate of delay for a size k job under any policy in SMART-LD,* $I_{\overline{W}}(k, m)$*, satisfies*

$$I_{V_{C-\epsilon}}(k, m) \leq I_{\overline{W}}(k, m) \leq I_{V_C}(k, m), \tag{4.3}$$

where $I_{V_\mu}(k, m)$ *is the virtual decay rate of delay under a priority queueing system, PRI, with capacity* $N\mu$ *and is defined as*

$$I_{V_\mu}(k, m) = \inf_{T \geq 0}\left[\inf_{\vec{z}:\mathcal{Z}}\left\{\mathfrak{A}_{<k}(\vec{z}) + \mathfrak{A}_k\right\}\right], \tag{4.4}$$

where condition $\mathcal{Z}$ *states that* $\sum_{i=1}^{k} iz_i = \mu(T + m + 1)$. *Further,*

$$\begin{aligned} \mathfrak{A}_{<k}(\vec{z}) &= \sum_{i=1}^{k-1} I_{A_i}^{(-T,m)}(z_i) \\ \mathfrak{A}_k &= I_{A_k}^{(-T,0)}(z_k). \end{aligned}$$

This theorem states that asymptotically (in the large capacity and large number of flows regime), *all policies in SMART-LD behave alike in terms of delay tail probability,* in that their delay decay rates are the same. In other words, for a job of original size k and for any fixed integer $m \geq 0$, we have that the delay distribution for $\overline{W}^{(N)}(k)$ is given by

$$P\left(\overline{W}^{(N)}(k) > m\right) = g(k, m)^N e^{-NI_{\overline{W}}(k,m)},$$

where $I_{\overline{W}}(k, m)$ *is the same for all policies in SMART-LD.* Thus, the decay rate of any policy in SMART-LD is the same as that of SRPT, which was derived in Chapter 3.

The decay rate in Equation (4.4) appears complicated, but does have intuition. It can be shown that in the many sources asymptote, the decay rate depends on the “most likely” way that the arrival processes deviate from their mean arrival rates in order to cause delay exceeding m. Thus, the two infimums choose the most likely time scale (T) and partition of the overall arrival rate to job sizes ($\vec{z}$) respectively. Then, inside the infimums, $\mathfrak{A}_{<k}(\vec{z})$ and $\mathfrak{A}_k$ characterize the delay caused to a size k job by jobs arriving over the time interval $(-T, m)$ with size $< k$ and by jobs arriving over the time interval $(-T, 0)$ with size k.

To illustrate this intuition, we now consider the special case when jobs are one of two sizes (1 or M). For this special case, we can provide simplified expressions which relate the delay distributions of jobs to the arrival process statistics. As a baseline for comparison, we compare with the delay distribution of FCFS [8].

For FCFS, it follows from the results in [8] that

$$\Pr(W^{(N)}(k) > m) \propto \max_{T\geq 0} \Pr(D^{(N)}(T) > C(T + m + 1)) \tag{4.5}$$

for $k = 1$ and $k = M$ and where $a \propto b$ means that a and b have the same decay rate. Here $D^{(N)}(T)$ is the cumulative workload (including both size 1 and size M jobs) that arrives to the server over the time-interval $(-T, 0)$, i.e.,

$$D^{(N)}(T) = A_1^N(-T, 0) + MA_M^N(-T, 0).$$

Recall that the server capacity is C units per time-slot. The above expression of Equation (4.5) states that the probability that a job of size 1 or M experiences a delay of at least m is the same as the probability that *the cumulative arrival workload over a time interval of* $T+1$ *slots exceeds the cumulative server capacity over a time interval of* $T+m+1$ *slots,* for some fixed value of T. The maximizing value of T in the right hand side (RHS) of Equation (4.5) is sometimes referred to as the critical time scale of the queue. Note that the decay rates for size 1 and M jobs are the same, since the only difference is their service requirement which is negligible in the large deviation framework.

Moving to SMART, it follows from Theorem 4.4.1 that

$$\Pr(W^{(N)}(k) > m) \propto \max_{T\geq 0} \Pr(E_k^{(N)}(T) > C(T + m + 1)) \tag{4.6}$$

where

$$E_k^{(N)}(T) \begin{cases} A_1^N(-T, 0), & k = 1; \\ A_1^N(-T, m) + MA_M^N(-T, 0), & k = M. \end{cases}$$

Note that for size 1 jobs, the above is the “best possible” delay distribution that can be achieved over the class of all work conserving policies. However, in the case of size k jobs for each fixed T, $E_M^{(N)}(T) \geq D^{(N)}(T)$, which immediately implies that the delay of a job of original size M with SMART-LD stochastically dominates (i.e., is larger in a distributional-sense) the corresponding delay with FCFS. However, for heavy-tailed arrivals, it can be shown that this difference is small, of order $O(1/M)$, by observing that the decay rate of SMART-LD matches that of SRPT, which has been compared to FCFS in Chapter 3. This means that in the large N and M regime (i.e. large number of flows, and a large difference in arriving job sizes), the delay experienced by a size M job is similar under FCFS and SMART-LD, while size 1 jobs experience far less delay under SMART-LD than under FCFS.

Proof. Intuition(technical proof in Appendix A.3): We show that the delay rate function of SMART-LD is again equivalent to the priority queue that gives higher priority to jobs of smaller original size, PRI. This is proven through the idea of *large scale limit*, similar to the arguments given for SRPT. We show that since at most only one job can be preempted at any time slot, preempted jobs are negligible compared to jobs that are not preempted. This idea is applied to the 2DQ framework of SMART-LD. As shown in Figure 4.2, the virtual queues are partitioned into areas of higher priority, lower priority, and no priority. The no priority area, B, is made up of exclusively preempted jobs, thus the number of jobs can be at most finite and can be effectively ignored in the large scale regime ($N \to \infty$). In a similar manner, we can see that the interiors of high and low priority areas are also sparse. This leads to the conclusion that in the many sources large deviations regime only the upper most strips are of importance. Upon further inspection, the upper strip of the two dimensional queue corresponds to PRI. $\square$

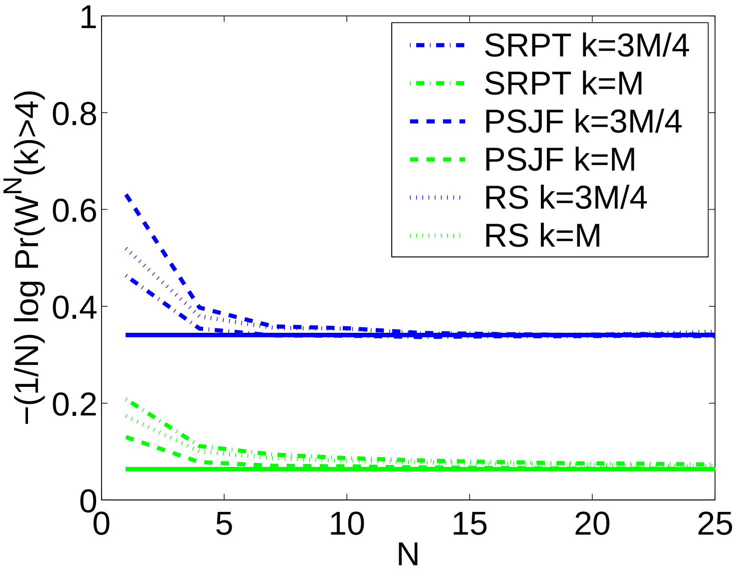
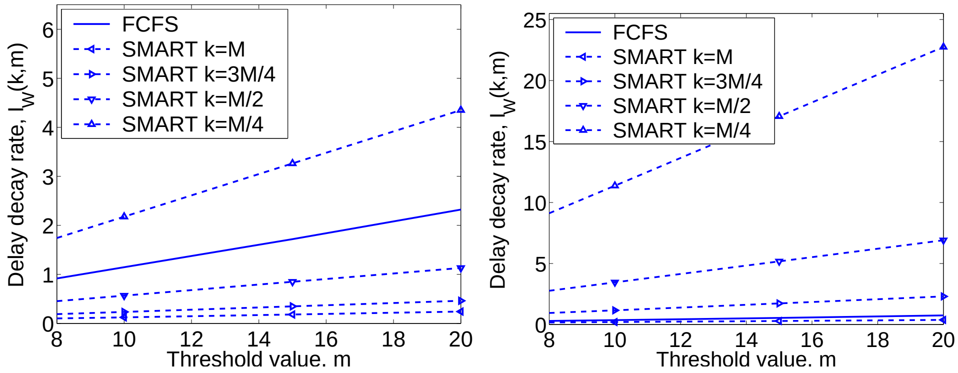


Figure 4.3: Plot of the rate of convergence of SRPT, PSJF, and RS to the decay rate under the uniform workload with $M = 16$, $\rho = 0.8$, and $m = 4$. The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation since a large delay for smaller job sizes is a very low probability event as N grows. Though not shown here, we found similar convergence rates under other policies in SMART-LD.

4.5 Numerical Analysis

Though Theorem 4.4.1 provides the expression for the delay decay rates of SMART-LD in the many sources regime, the complicated nature of these formulas hide the behavior of the decay rates. In this section, we will use simulations and numerical experiments to illustrate how the decay rate $I_W(k,m)$, and thus $Pr(W(k) > m)$, is affected by the variability of the service distribution, the range of the service distribution (M), and the threshold value (m). In performing these studies we focus on three practical questions:

(i) How does the delay distribution behave under a large, but not infinite, number of flows? That is, what is the rate of convergence to the decay rate, $I_W(k,m)$?



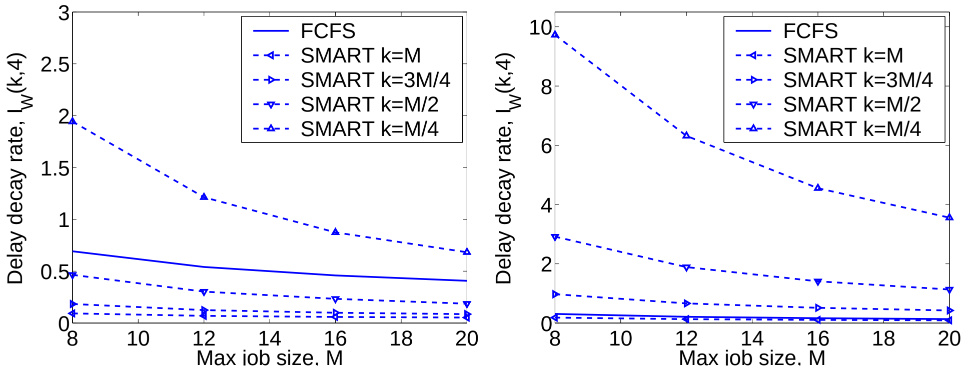
(a) Power-law (b) High variability

Figure 4.4: Plot of the decay rate as a function of the threshold m under the power-law and the high variability workload under SMART-LD with the maximum job size $M = 16$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted.

(ii) How does the decay rate for a job of size k vary across k? That is, how much do large job sizes suffer under policies that bias towards small job sizes?

In our experiments, we assume that jobs are of sizes 1, $M/4$, $M/2$, $3M/4$, or M, and we vary M between 8 and 20. Each job size arrives according to an on-off process, i.e. in each discrete interval a size k job arrives with probability p_k. We assume that the capacity of the system, C, is 1.

We will consider three cases for the distribution of job sizes, which we refer to as *uniform*, *power-law*, and *high variability*. In the *uniform* case each job size is equally likely. In the *power-law* case, the arrival probabilities follow the power-law distribution with exponent 2, i.e. a discrete and truncated counterpart of the Pareto distribution. Note that due to the small spread of job sizes, this distribution is not highly variable; thus, to study the impact of variability, we also consider a distribution where the largest jobs make up half the load (as has been observed in web job sizes). In particular, the *high variability* workload has size 1, $M/4$, $M/2$, $3M/4$, and M job arrivals make up 1/24, 1/12, 1/8, 1/4,



(a) Power-law

(b) High variability

Figure 4.5: Plot of the decay rate as a function of the maximum job size M under the power-law and the high variability workload under SMART-LD with the threshold $m = 4$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted.

and 1/2 of the total load.

Figure 4.3 illustrates the convergence of the delay distribution to the asymptotic decay rate as N grows under SRPT. The thick lines are the numeric calculations of the asymptotic decay rates proven in the dissertation and the other dotted lines are generated using an event-driven simulation for scheduling policies SRPT, PSJF, and RS (which prioritizes towards the smallest product of remaining size and original size) that are all included in SMART. The simulation matches the uniform workload and the setup described above except that a Poisson arrival process (for ease of simulation) is used. Thus, in addition to the error from using a finite N, Figure 4.3 illustrates the error from the discretization of the arrival process. Note that when $N = 20$ there is already little difference between the empirical decay rate of SRPT, PSJF, and RS and the asymptotic limit of SMART-LD. Thus, it seems that rate of convergence to the decay rate is very fast, and thus the asymptotic decay rate provides information that is useful in practical settings such as high traffic web servers and routers, which have far more than 20 simultaneous flows.

We investigate the effect of varying m and M under the power-law and the high variability job size distribution on the delay decay rate and we illustrate the effect of these variables in Figures 4.4 and 4.5 respectively. These plots illustrate the effect of increasing m and M on the decay rate of delay. As the threshold value m increases, Figure 4.4 shows that the decay rate increases, and thus $Pr(W(k) > m)$ decreases. It is interesting to note that the decay rate seems to grow linearly with m for all jobs sizes k under SMART-LD. As the maximum job size M increases, Figure 4.5 shows that the decay rate of all job sizes decreases, which is not surprising since this leads to an increase in service times for all job sizes.

Chapter 5

LAS

5.1 Background and Related Work

An implementation issue with SRPT or SMART is that the remaining processing time information required may not be available to the scheduler. For example, a request of a web page may entail downloading or searching other web sites for files of unknown size. However, even in applications where the job sizes are unknown, system designers have suggested policies such as Least-Attained-Service (LAS), which prioritizes jobs with small attained service so that small jobs (which always have small ages) tend to have the server to themselves. Formally, LAS is defined as follows.

Definition 5.1.1. *LAS is a preemptive scheduling policy that serves jobs with the smallest attained service (age) first. When there are multiple jobs with the same attained service, the capacity of the server is distributed among them in some manner.*

Note that a newly arriving job always preempts the job (or jobs) currently in service and retains the processor until one of the following occurs: (i) the job departs, (ii) the next arrival appears, or (iii) the job has obtained an amount of service equal to that received by the job(s) preempted on arrival.

More specifically, in routers where the remaining processing time is not known many blind[1] policies have been suggested. Of such policies, Least–Attained–Service (LAS) has been shown to be optimal in mean delay when the job size distribution has the decreasing failure rate property [35, 36], and many variants have been implemented in practice [32, 33] with similar success as SRPT. It has been shown that the amount of service a job has received so far is a good indication of its remaining processing time when the jobs size distribution follows a decreasing failure rate. Job size distribution with decreasing failure rate captures the actual job size distribution found in the Internet since the well known heavy tail job size distribution that is known to accurately describe the actual job size distribution in today's computer systems have a decreasing failure rate. By assigning higher priority to jobs with small attained service, LAS indirectly favors jobs of small original size, which always have small attained service. Due to this property, LAS is viewed as a good approximation to SRPT when the job size information is not available.

Similar to SRPT and SMART, recent studies on LAS have focused on the mean delay experienced by a job of size k, $E[W(k)]$[17, 32, 33, 41]. However, when one is concerned with other metrics such as QoS in a more realistic setting, a more revealing metric would be the *distribution* of $W(k)$ in the many sources large deviations regime.

The difficulty in direct analysis of the distribution of $W(k)$ has led researchers to study asymptotic scalings of the distribution. Analytical results on the LAS policy in the *large buffer* large deviations framework have shown that the tail of $W(k)$ behaves proportionally to a busy period, when the job size distribution is truncated at k. In other words, jobs of size larger than k contribute k to the busy period [25, 30, 31]. This is in contrast to the behavior of FCFS, where the tail of $W(k)$ is proportional to the tail of the stationary workload for all k. However, the many sources large deviations regime allows us to understand complementary characteristics of the LAS policy in a setting which is geared to understand today's web-server.

---

[1] Blind to job size.

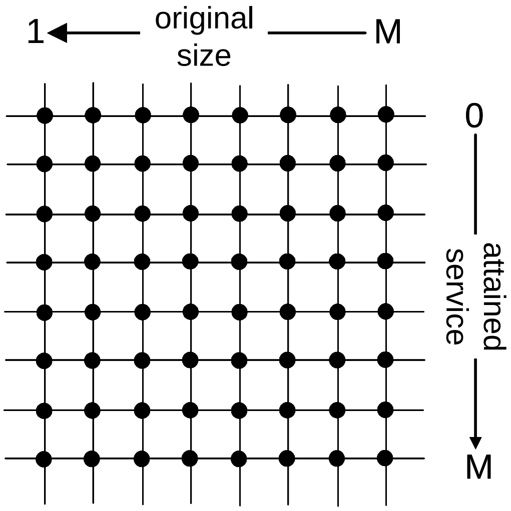
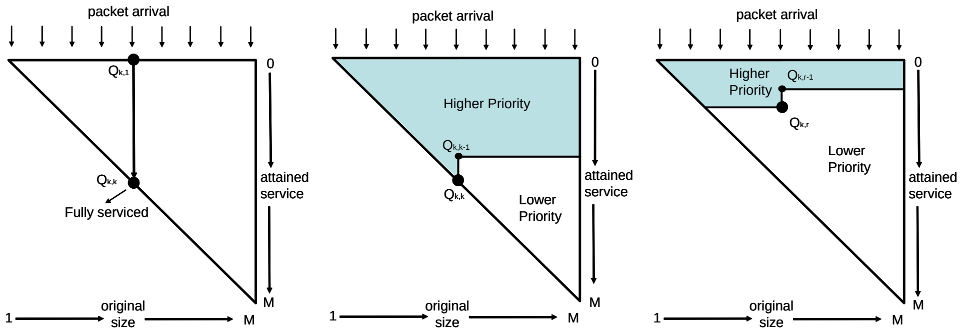


Figure 5.1: Illustration of the 2DQ representation for LAS. As depicted the X-axis is the original size and the Y-axis is the attained service of a job, where attained service is the amount of service a job has received so far.

In this chapter, we analyze the delay distribution of LAS in the many sources large deviation regime which to the best of our knowledge has not been studied yet.

5.2 Intuition

The analysis of LAS in the many sources large deviations regime hinges on the 2DQ framework studied in Section 4.2. However, the arrangement of virtual queues for LAS is different from SMART-LD due to its prioritization with respect to attained service rather than the remaining processing times as in SMART-LD. The 2DQ framework for LAS is depicted in Figure 5.1, where the X-axis is the original size of a job and Y-axis is the attained service that a job has received so far. This may seem surprising since the prioritization of LAS depends only on the attained service of a job; however adding a secondary variable is the key to making the analysis tractable. Note that the analysis of LAS must be very different from SRPT or SMART, since a job actually loses priority as it receives service while for SRPT and SMART a job gains priority.



(a) (b) (c)

Figure 5.2: Illustrations of the two dimensional queueing framework for LAS. Note that since a job cannot have attained service larger than its original size only the upper right triangle of Figure 5.1 is of importance. The progression of a job between queues while in the system is illustrated in (a). The priority structure for an incoming job is shown in (b) and for a partially served job is shown in (c).

As shown in Figure 5.1, the secondary variable we add is the original job size. This means that, in the event of ties in attained service, instead of sharing the server among the jobs with equal attained service, the job with the smallest original size is served first. Further, jobs that have the same original size and the same attained service are serviced according to a FCFS rule. Keep in mind that jobs are not served with the full service capacity, but are only serviced a single unit at once, which is consistent with the discrete version of LAS. Note that using the job size as a secondary ordering variable does not alter the performance of LAS in the asymptotic framework, it is simply a modeling decision used to make the analysis tractable.

Formally, a queue $Q_{i,j},\ i \geq j$, denotes the queue that contains all the jobs having original size i that have received j unit of service. Thus a job of original size k arrives to $Q_{k,0}$ and then progresses to $Q_{k,1}$, $Q_{k,2}$ ... $Q_{k,k-1}$, $Q_{k,k}$, at which point the job is fully serviced and leaves the system. This is depicted in Figure 5.2. We again denote $Q_{i,j}(t)$ as the volume of $Q_{i,j}$ at time t, and $Q_i(t)$ as the volume of the queue that contains all jobs that have original size i, Q_i, where $Q_i = \bigcup_{j=0}^{j=i-1} Q_{i,j}$.

Let us now consider a tagged size k job that arrives in the system at time 0 and has received r units of service (see Figure 5.2). By the definition of LAS, all jobs that have been served less than r units have higher priority than the tagged job. Additionally, jobs with smaller original size that have attained r units of service also have higher priority. In other words, for any job in $Q_{k,r}$ to be serviced an additional unit, the following queues must be empty.

$$\bigcup_{i=1}^{k-1}\bigcup_{j=0}^{r} Q_{i,j} + \bigcup_{i=k+1}^{M}\bigcup_{j=0}^{r-1} Q_{i,j} + \bigcup_{j=0}^{r-1} Q_{k,j}$$

Further, since in each queue, $Q_{i,j}$, jobs are serviced in a FCFS order, jobs that have arrived to Q_k before the tagged job must be serviced r units and job that arrived after must be served only $r-1$ units. The same argument can be made for $Q_{k,k-1}$, which is the queue for jobs that will leave the system once it is served again. This group of higher priority queues change as the job in question receives service. However, the basis of the derivation of the delay decay rate rests in the fact that these higher priority queues follow a tractable and consistent order.

5.3 Main Contributions

The result of this chapter in the dissertation is the derivation of the delay decay rate of LAS and a corollary that compares the asymptotic delay distribution of SMART-LD and LAS across job sizes. The following theorem describes the delay rate function of LAS in the many sources large deviations regime, i.e., the asymptotic delay distribution of LAS.

Theorem 5.3.1. *Under similar assumptions made in Section 4.4.2, the decay rate of delay for size k jobs under LAS, $I_{\hat{W}}(k,m)$, is*

$$I_{\hat{W}}(k,m) = \inf_{T\geq 0}\left[\inf_{\vec{y}:\mathcal{Y}}\left(\mathfrak{A}_{<k}(\vec{y}) + \mathfrak{A}_{k}(\vec{y}) + \mathfrak{A}_{>k}(\vec{y})\right)\right], \tag{5.1}$$

where condition $\mathfrak{Y}$ *states that* $\sum_{i\in\hat{\mathbf{k}}} iy_i + ky_k + \sum_{i\in\check{\mathbf{k}}}(k-1)y_i = C(T+m+1)$ *with* $\hat{\mathbf{k}}\{1,\ldots,k-1\}$, $\check{\mathbf{k}} = \{k+1,\ldots,M\}$*, and* $y_k^{(1)} + \frac{k-1}{k}y_k^{(2)} = y_k$*. Further,*

$$\begin{aligned}
\mathfrak{A}_{<k}(\vec{y}) &= \sum_{i=1}^{k-1} I_{A_i}^{(-T,m)}(y_i) \\
\mathfrak{A}_{k}(\vec{x}) &= I_{A_k}^{(-T,0)}(y_k^{(1)}) + I_{A_k}^{(1,m)}(y_k^{(2)}) \\
\mathfrak{A}_{>k}(\vec{y}) &= \sum_{j=k+1}^{M} I_{A_i}^{(-T,m)}(y_j).
\end{aligned}$$

Theorem 5.3.1 characterizes the delay distribution of LAS in the many sources regime. Though the form of Equation (5.1) is complicated, we can obtain intuition for it. Again, the decay rate depends on the “most likely” way that the arrival processes deviate from their mean arrival rates in order to cause delay exceeding m. Thus, the two infimums choose the most likely time scale (T) and arrival rates for each job size ($\vec{y}$), where y_k is separated into the arrivals before ($y_k^{(1)}$) and after ($y_k^{(2)}$) the tagged arrival. Then, inside the infimums, $\mathfrak{A}_{<k}(\vec{y})$, $\mathfrak{A}_{k}(\vec{y})$, and $\mathfrak{A}_{>k}(\vec{y})$ characterize the contribution to the delay of a size k job made by jobs with size $< k$, other jobs of size k, and jobs of size $> k$ arriving in the time interval $(-T, m)$. This intuition indicates one key difference between the decay rates of $W(k)$ under SMART and LAS. While under LAS $\mathfrak{A}_{>k}(\vec{y})$ characterizes the effect of jobs with size larger than k, there is no such term in the decay rate of SMART (see Equation (4.4)).

To illustrate the intuition of the above result, we will use the special case when jobs are only of sizes 1 and M. For LAS, in the special case when there only exist size 1 and M jobs, it follows from Theorem 5.3.1 that

$$\Pr(W^{(N)}(k) > m) \propto \max_{T\geq 0} \Pr(F_k^{(N)}(T) > C(T+m+1)) \tag{5.2}$$

where

$$F_k^{(N)}(T) = \begin{cases} A_1^N(-T, 0), & k = 1; \\ A_1^N(-T, m) + MA_M^N(-T, 0) & \\ \qquad +(M-1)A_M^N(1, m), & k = M. \end{cases}$$

Note, that for size 1 jobs, SMART-LD and LAS are asymptotically identical (i.e., the decay rates are the same). On the other-hand, for size M jobs, observe that for each fixed T, $F_M^{(N)}(T) \geq E_M^{(N)}(T)$, which immediately implies the delay of a job of original size M with LAS stochastically dominates the corresponding delay with SMART-LD. Thus, the delay experienced by a size M job under LAS is larger than that under SMART-LD (in distribution). This is an important observation that will be generalized in Corollary 5.3.1.

Proof. Intuition(technical proof in Appendix A.4): As shown in Figure 5.2, for LAS the two dimensional queueing framework partitions the virtual queues into coherent areas of higher and lower priority with respect to a job in the system. The job progresses downward through the two dimensional queue only when all the queues in the higher priority area are emptied. This logical separation of high and low priority set of virtual queues persists in a tractable manner until the job leaves the system as shown in Figure 5.2. In this manner, LAS can be partitioned into two distinct areas of high and low priority that is time varying. This observation along with the well known results from the priority queueing system are applied to derive the delay rate function of LAS □

In addition to the delay rate function of LAS, we prove that all policies in SMART-LD stochastically outperform LAS with respect to delay for any job size [48], i.e., there is a consistent penalty for not using the remaining processing time directly which affects all job sizes. We can prove the following corollary using a simple extension of the size 1 and M example.

Corollary 5.3.1. *Any scheduling policy in SMART is uniformly better (for any job size)*

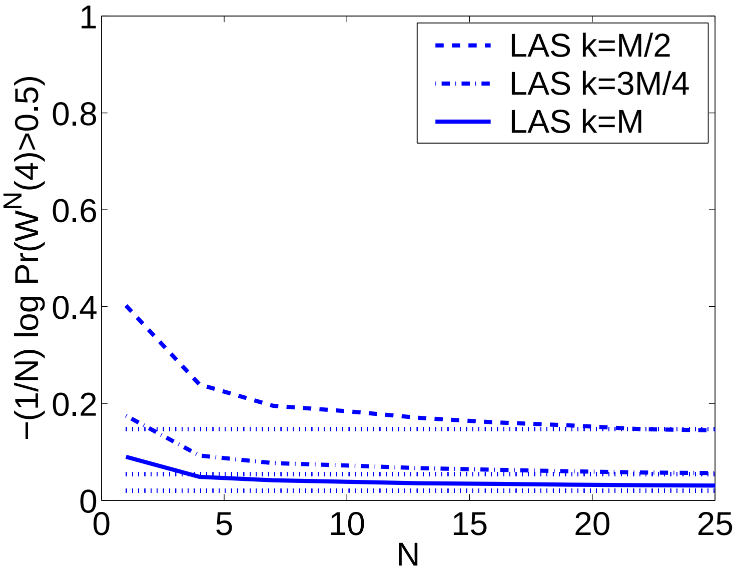


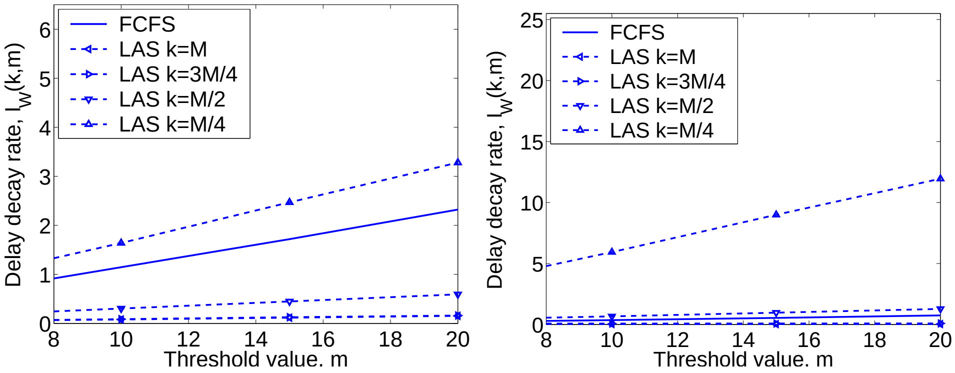
Figure 5.3: Plot of the rate of convergence of SRPT to the decay rate under the uniform workload with $M = 16$, $\rho = 0.8$, and $m = 4$. The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation since a large delay for smaller job sizes is a very low probability event as N grows. Though not shown here, we found similar convergence rates under other policies in SMART-LD.

than the LAS policy with respect to delay in the many sources large deviations regime, i.e.

$$I_{\overline{W}}(k,m) \geq I_{\hat{W}}(k,m),$$

for all k and m, where $I_{\overline{W}}(k,m)$ *and* $I_{\hat{W}}(k,m)$ *are the delay rate functions of SMART and LAS respectively.*

Proof. The precise proof for Corollary 5.3.1 is omitted. However, the basic idea behind the proof is that the volume of higher priority jobs for SMART-LD is always less than that of LAS while the available capacity is the same for both. In particular, $\mathfrak{A}_{<k}(\vec{z}) + \mathfrak{A}_k$ of SMART-LD is always less than $\mathfrak{A}_{<k}(\vec{y}) + \mathfrak{A}_k(\vec{y}) + \mathfrak{A}_{>k}(\vec{y})$ of LAS (see Theorem 4.4.1 and Theorem 5.3.1 $\square$



(a) Power-law (b) High variability

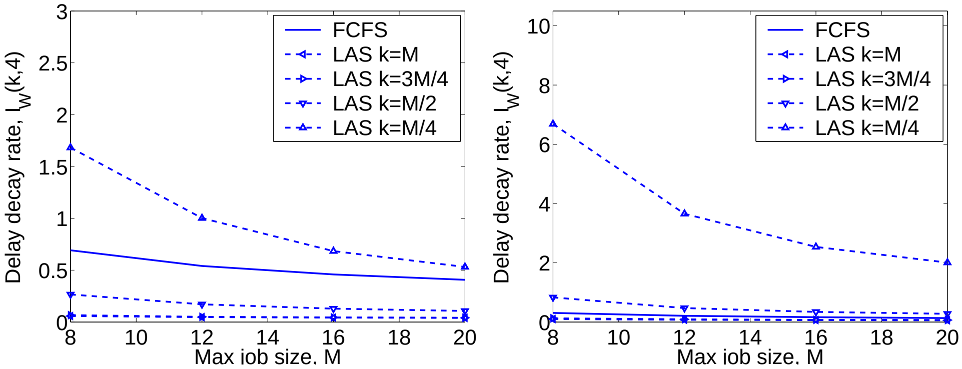
Figure 5.4: Plot of the decay rate as a function of the threshold m under the power-law and the high variability workload under SMART-LD with the maximum job size $M = 16$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted.

5.4 Numerical Analysis

Similar to the SMART-LD case, we address the same questions as well as the following additional question.

(i) How much penalty does LAS pay for not using job size information to prioritize? That is, by how much does SMART-LD outperform LAS?

All the simulation setup used in this section is identical to the SMART-LD case. Figure 5.3 illustrates the convergence of the delay distribution to the asymptotic decay rate as N grows under LAS. The dotted lines are the numeric calculations of the asymptotic decay rates proven in the dissertation and the other lines are generated using an event-driven simulation. The simulation matches the uniform workload described above except that a Poisson arrival process is used. Thus, it seems that rate of convergence to the decay rate is very fast, and thus the asymptotic decay rate provides information that is useful in practical settings such as high traffic web servers and routers, which have far more than 20



(a) Power-law (b) High variability

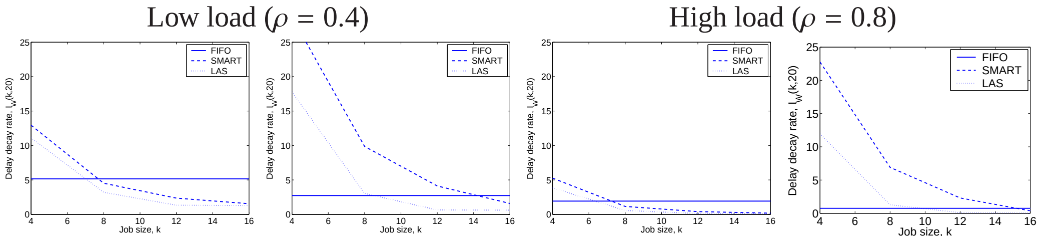
Figure 5.5: Plot of the decay rate as a function of the maximum job size M under the power-law and the high variability workload under SMART-LD with the threshold $m = 4$ and $\rho = 0.8$. Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. The decay rate of FCFS is included as a benchmark. Note that since decay rates of size 1 jobs are infinite, they are omitted.

simultaneous flows.

We investigate the effect of varying m and M under the power-law and the high variability job size distribution on the delay decay rate and we illustrate the effect of these variables in Figures 5.4 and 5.5 respectively. Figure 5.4 shows that the decay rate increases, and thus $Pr(W(k) > m)$ decreases, for increasing m. As the maximum job size M increases, Figure 5.5 shows that the decay rate of all job sizes decrease.

More importantly, we address the additional question (iii). Figure 5.6 illustrates how the decay rate for a job of size k varies across k under SMART-LD, LAS, and FCFS (which we include as a baseline for comparison). The results are shown for both the power-law and high variability workloads under high load. *Note that a larger decay rate indicates a stochastically smaller delay.*

The first observation we make is that, in each of the plots, small job sizes have much better decay rates under SMART-LD and LAS than under FCFS; whereas large job sizes have better decay rates under FCFS than under LAS and SMART-LD. Thus, there is always crossover point for each of SMART-LD and LAS where their decay rate "crosses over" that



(a) Power-law (b) High variability (c) Power-law (d) High variability

Figure 5.6: Plot of the delay rate function as a function of the job size, k, with the threshold $m = 20$ and maximum job size $M = 16$ held fixed. Recall that $I_W(k, m)$ measures the rate function of $Pr(W(k) > m)$ and that a larger $I_W(k, m)$ indicates a stochastically smaller delay.

of FCFS. Figure 5.6 illustrates that the crossover point is highly dependent on the service distribution. When the load is high and the largest jobs make up a significant fraction of the load, the decay rate of SMART-LD does not cross that of FCFS until the largest job size, $k = M$. The behavior of the crossover point can be understood using the following key observation: while the decay rate under FCFS gets worse (smaller) as the load of largest jobs is increased and the total load is held constant, the decay rates of LAS and SMART-LD get better (larger). Thus, since large job sizes make up a significant fraction of the load in many computer applications, it seems that one need not worry too much about the suffering of large job sizes under policies that prioritize small jobs. Though not shown, we also investigated the impact of load on the crossover point and found that load only changes the magnitude of the decay rates (the higher the load, the lower the decay rate), not the relative behavior of the decay rates under FCFS, LAS, and SMART-LD.

The next observation we make from Figure 5.6 is that both SMART-LD and LAS exhibit a similar trend in decay rate across k, with SMART-LD always providing stochastically smaller delays than LAS. Further, in answer to the question, Figure 5.6 illustrates that the improvement of SMART-LD over LAS is again highly dependent on the job size distribution: as the load of the largest jobs increases, the difference in the decay rates of SMART-LD and LAS increases. The fact that SMART-LD is better for small jobs follows from the operation of two policies: small jobs typically do not get preempted under SMART-LD, but are preempted by all arrivals under LAS. However, the result that SMART-LD is better than LAS even for larger jobs less obvious. An explanation for this fact is that under LAS, though larger jobs gain higher priority at arrival compared to SMART-LD, as large jobs receive service their priority is dropping quickly under LAS but may be increasing under SMART-LD.

Chapter 6

finite-SRPT

6.1 Background and Related Work

As we have discussed in Chapter 3, SRPT is a policy that provides superior delay properties for smaller jobs while the unfairness for larger jobs is not large. However, one problem in implementing SRPT is practical systems is that SRPT does not take into consideration the bandwidth constraint at links. In other words, SRPT guarantees full bandwidth (BW) of the server to job requests with the smallest remaining processing time until the request are fully accommodated or are preempted. This is unrealistic when we consider servers of large BW. To illustrate, let us consider the following example. Assume that the outgoing BW of the web-server is 1 Gbps and multiple download requests for file downloads arrive. The original SRPT algorithm at the web-server would process such requests one at a time with the full 1 Gbps out-going link bandwidth allocated to the file currently being served. However, the end-receiver (user) receiving this file may have bandwidth of only 10 Mbps (e.g., a wireless LAN at the end-user). Thus a more realistic scenario is where the BW of the server is simultaneously distributed among multiple file requests by a fixed amount that corresponds to the link rates of the end-receivers. This can be captured by imposing an additional constraint on SRPT – namely, a peak rate at which each file can be served by the web-server.

In this section, we investigate the delay distribution of SRPT with the additional constraint that the maximum amount of service each job can receive in a time-slot (also called the potential service) is bounded (this captures the *end-receiver* bandwidth constraint). We denote the proposed scheduling policy as finite-SRPT. We show that as the maximum potential service a job can receive in a time-slot is varied, the proposed finite-SRPT exhibits a spectrum of characteristics. In particular, when the maximum amount is larger than or equal to the largest job it is equivalent to SRPT whereas it is similar to Processor-Sharing (PS) when the maximum amount is only one unit.

6.1.1 Definition and explanation of finite-SRPT

SRPT is a preemptive scheduling policy that serve jobs with smaller remaining processing time first. Among jobs that have the same remaining processing time, jobs are served in FCFS order[1]. We consider SRPT for a more realistic setting where the jobs are served at most a fixed amount in a time-slot. Accordingly, we call this realistic version of SRPT as finite-SRPT.

Definition 6.1.1. ***finite-SRPT*** *is identical to the original SRPT but with an additional restriction which states that a job is served at most* D *units in a time slot, where* $1 \leq D \leq M$ *(recall* M *is the largest job size).*

Next section explains the proposed finite-SRPT with respect to the two-dimensional-queueing (2DQ) framework first introduced in [48] which makes the analysis possible.

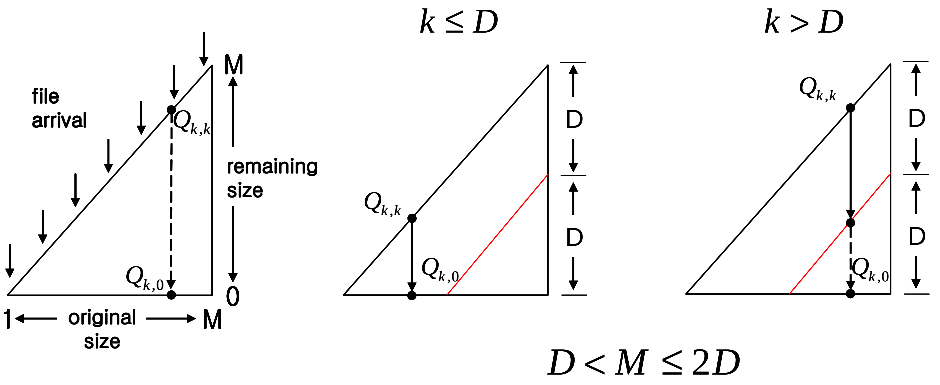


Figure 6.1: Finite-SRPT and two dimensional queueing framework. The left-most figure depicts the path that a job should take for it to be fully serviced. X-axis is the original size of a job and Y-axis is the remaining size of a job. Thus a job of original size k upon receiving service progresses downward until it is fully served, i.e., remaining processing time is 0. However, different sized jobs require different numbers of rounds of D units of service in finite-SRPT. jobs of size $k \le D$ will be fully served in one round, while jobs with size $D < k \le 2D$ require two rounds of service to leave the system.

6.2 Intuition

As in Chapter 4, we consider a 2DQ representation, and let $Q_{i,j}$ denote the fictitious queue that contains all the jobs that were originally of size i and currently have remaining size j. Recall that our objective is to study the delay experienced by the tagged job – a fictitious job that arrived to the system at time-slot 0. The path that the tagged job of size k takes in the 2DQ representation when $D < M \le 2D$ is depicted in Figure 6.1. Accordingly the 2DQ representation allows the separation of high and low priority areas with respect to the tagged job as shown in Figure 6.2.

We first define a typical job for the finite-SRPT scheduling policy as follows.

Definition 6.2.1. ***Typical jobs*** *are jobs that receive D units of service every time they are selected for service. Observe that in a time-slot in which a job is selected for service, it can*

---

[1] Other definition of SRPT specifies that jobs with the same remaining size are served in a PS manner. Even in such cases, the system does not restrict the amount of service to a job since the number of jobs with the smallest remaining size is not fixed, i.e., can range from 1 job to all jobs in the system.

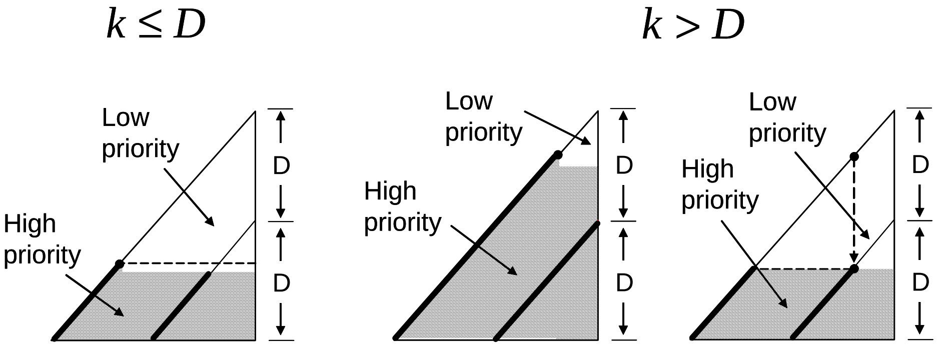


Figure 6.2: Priority scheme for a tagged job of size k, where $D < M \leq 2D$. As depicted in the figures, all fictitious queues in the shaded area are of higher priority. Correspondingly, lower priority queues are the fictitious queues in the non-shaded area. The higher priority area is composed of typical higher priority jobs represented as the thick diagonal lines and atypical jobs are in the rest of the shaded area.

receive at most *D units of service. We denote a job to be typical if it is offered* exactly *D units of service each time it is selected for service. Thus a typical job receives D units of service until its leftover size becomes less than D whence it is fully served. A job is said to be* ***atypical*** *if it is not typical.*

To illustrate, consider a system with link capacity $C = 10$, with 2 jobs (A,B) in the system of sizes $F_A = 16$, $F_B = 24$ respectively and assume $D = 8$. In this case, the state of the system at the end of service is $F_A = 8$, $F_B = 22$, i.e., jobs A, B received 8, 2 units of service respectively. In this example, job A is typical and job B is atypical.

To illustrate the path that the tagged job of size k takes, we consider the specific case of $D < M \leq 2D$. In this example (and in the rest of this section), we assume that the all jobs (including the tagged job) are typical[2].

The tagged job takes two distinct paths depending on its size, i.e., $k \leq D$ and $D < k \leq 2D$, since they require 1 round and 2 rounds of service to completely leave the system. Denote T_1 as the time-slot in which the tagged job receives its first D units of service, and

---

[2] Note that in general, some of the jobs could be atypical – this problem will be dealt in the proof of the delay decay rate.

T_2 as the time-slot in which the tagged job receives its second D units of service (if the tagged job size exceeds D).

First, let us consider the path of the tagged job of size $k \leq D$, and in the process identify all higher priority queues and arrivals that are required to derive the virtual delay experienced by the tagged job. This tagged job arrives at $Q_{k,k}$ at time-slot 0 by definition. Since all jobs with smaller remaining size have higher priority, all of the jobs in the following queues must be served D units before the tagged job is served its first D units.

$$\sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0). \tag{6.1}$$

Since $k \leq D$, the queues in Equation (6.1) must be fully served before the tagged job leaves the system. Note that since $k \leq D$, the tagged job leaves the system at time-slot T_1. Thus, in addition to the jobs in the higher priority queues described in Equation 6.1, job arrivals $\sum_{i=1}^{k-1} A_i^N(T_1, 1)$ must be fully served (since $k \leq D$) before the tagged job leaves the system.

Now, we consider the path that the tagged job of size $D < k \leq 2D$ takes and determine the higher priority queues and job arrivals using the 2DQ framework. At time-slot 0, the tagged job arrives at $Q_{k,k}$. Thus all jobs in Equation (6.1) must be served D units in order for the tagged job to receives its first D units of service. Since the tagged job receives its first D units of service at T_1, job arrivals $\sum_{i=1}^{k-1} A_i^N(T_1, 1)$ must be served D units. However, the tagged job requires another D units of service to leave the system. In fact, the tagged job will be in the queue $Q_{k,k-D}$ at time T_1. Thus, jobs in the following queues:

$$\sum_{i=1}^{M} \sum_{j=1}^{k-D-1} Q_{i,j}(T_1) + Q_{i,k-D}(T_1) \tag{6.2}$$

must be served D units before the tagged job receives its next D units of service. Note that all jobs in Equation (6.2) are ones that have been served D units from the queues described in Equation (6.1) and arrivals $\sum_{i=1}^{k-1} A_i^N(T_1, 1)$. Note again that since the jobs in Equation (6.2) and $\sum_{i=1}^{k-1} A_i^N(T_1, 1)$ at time-slot T_1 have remaining size $< D$, they need to be fully served before the tagged job receives its second D units of service. In addition, all new job arrivals that have original size $< k - D$, i.e., $\sum_{i=1}^{k-D-1} A_i^N(T_2, T_1 + 1)$, are required to be served D units (since all jobs in this arrival have original size $< D$, they are fully served) before the tagged job can leave the system. To summarize, all jobs in the queues described in Equation (6.1) and job arrivals $\sum_{i=1}^{k-1} A_i^N(T_1, 1)$ and $\sum_{i=1}^{k-D-1} A_i^N(T_2, T_1 + 1)$ should be fully served before the tagged job can leave the system.

This argument can be extended to the general case of a tagged job of size $(i-1)D < k \leq iD$ for some integer $i \geq 1$.. In this case, the tagged job requires i rounds of service of being serve D units to be fully served. The higher priority queues and arrivals can be determined by defining T_i as the time-slot when the tagged job receives its ith D unit of service.

6.3 Main Contributions

6.3.1 Characteristics of finite-SRPT

One can see that finite-SRPT behaves differently for different values of D, which ranges from 1 to M. One extreme case is when $D = M$. Finite-SRPT for $D = M$ is a scheduling policies that does not restrict the amount of service granted to a job in a time-slot. Thus finite-SRPT with $D = M$ allocates its full capacity to a single job then moves on to the next one, which is equivalent to the original SRPT.

As D decreases, finite-SRPT becomes a scheduling policy where a job is served less and less each time it is scheduled for service. When $D = 1$, a job is served a unit at each time-slot it is selected for service. Define a cycle to be the interval between time-slots when the tagged job is served D units and the time-slot it is served D units again, i.e.,

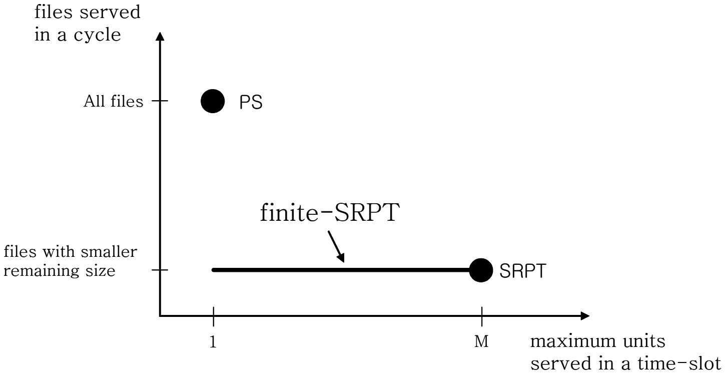


Figure 6.3: Finite-SRPT described with respect to SRPT and PS. Finite-SRPT occupies a region between SRPT and PS for varying values of D. When $D = M$, finite-SRPT is equivalent to SRPT. Finite-SRPT at $D = 1$ is a scheduling policy very close to PS but different in the fact that only jobs with smaller remaining processing time are served before the tagged job in a cycle instead of all jobs in the system.

$(T_{i+1}, T_i + 1)$. Then finite-SRPT with $D = 1$ is a scheduling policy which serves *all jobs with smaller remaining size* 1 unit in a cycle. This is very similar to a discrete version of PS where *all jobs* are served 1 unit in a cycle. As explained, the only difference lies in the fact that the two policies differ in the set of jobs that is served 1 unit in a cycle (all smaller remaining sized jobs vs. all jobs). This interesting characteristic of finite-SRPT is due to the fact that when the maximum service (D) is small, the left-over capacity after all smaller jobs have been served D units trickles down to larger jobs thus making the scheduling policy more fair.

For example, consider the following scenario where the system has three jobs of size 3, 4, and 10. Assume that the capacity of the server is 3. In the original SRPT or when $D = 3$ in finite-SRPT, only the first job of size 3 will be served in the first time slot. However, when $D = 2$, 2 units the first job and 1 unit of the second job will be served. Note that a larger job which would not have been served if it was SRPT is being served. In other words, left-over capacity is distributed to the larger jobs. This trend becomes more pronounce when we consider $D = 1$ for finite-SRPT. In this case, all three jobs will receive 1 unit of service which is what PS would have done. However, note that finite-SRPT becomes equivalent to PS only when the number of jobs is smaller than $\lfloor \frac{totalcapacity}{D} \rfloor$.

Intuitively, the proposed finite-SRPT can be seen as a scheduling policies that span the area shown in Figure 6.3 as D is varied. When $D = M$, finite-SRPT is equivalent to SRPT and as D decrease finite-SRPT becomes similar to PS. This intuition can be explicitly seen in the numerical analysis in Section 6.4.

6.3.2 The delay decay rate of finite-SRPT

The delay decay rate of finite-SRPT for varying D in the many sources large deviations regime is derived in the following theorem [45].

Theorem 6.3.1. *Under suitable straight line large deviation and burstiness assumptions on the arrival (used in [44]), the virtual delay decay rate of size k jobs under finite-SRPT satisfies*

$$I^{C-\epsilon}(k,m) \leq I_V(k,m) \leq I^{C+\epsilon}(k,m) \tag{6.3}$$

for any $\epsilon > 0$, where

$$I^S(k,m) = \inf_{\vec{T}:\mathcal{T}} \left[\inf_{\vec{y}:\mathcal{Y}} \left\{ \sum_{j=1}^{M} I_{A_j^N(T_{i+1},T_i+1)} \left(y_j^{(T_{i+1},T_i+1)} \right) \right\} \right], \tag{6.4}$$

for which the condition $\mathcal{T}$ states that $m = T_{\lceil \frac{k}{D} \rceil - 1} > \ldots > T_1 > 0 > T_0 > \ldots > T_{\lceil \frac{k}{D} \rceil - \lceil \frac{M}{D} \rceil}$, and $\mathcal{Y}$ states that

$$\sum_{i=1}^{\lceil \frac{k}{D} \rceil - 1} \sum_{j=1}^{k-iD-1} j y_j^{(T_{i+1},T_i+1)} + \sum_{j=1}^{k-1} j y_j^{(T_1,1)} + \sum_{j=1}^{k} j y_j^{(0,T_0+1)}$$

$$+\sum_{i=\lceil \frac{k}{D}\rceil-\lceil \frac{M}{D}\rceil}^{-1}\sum_{j=1}^{k-iD} jy_j^{(T_{i+1},T_i+1)} = S\left(T_{\lceil \frac{k}{D}\rceil-\lceil \frac{M}{D}\rceil}+m+1\right),$$

where $\lceil a\rceil$ *is the smallest integer large than* a*, and* $y_j^{(T_{i+1},T_i+1)}, y_j^{(T_1,1)}, y_j^{(0,T_0+1)} \geq 0$ *for all* $i, j \geq 0$.

The theorem states that the delay decay rate of finite-SRPT can be bounded by the delay decay rate of finite-SRPT considering only typical jobs with slight perturbations (ϵ) to the server capacity.

Proof. Intuition(technical proof in Appendix A.5): The basic idea behind the proof is the following. The derivation of the delay decay rate for finite-SRPT is dependent upon identifying the critical event that results in $\{V^{(N)}(k) > m\}$ with the least cost. This critical event corresponds to the "easiest" way in which the tagged job of size k arriving at time-slot 0 does not leave the system by time-slot m. We make the observation that the "easiest" way to achieve $\{V^{(N)}(k) > m\}$ is the event in which all higher priority jobs are serviced before the tagged job.

However, the precise volume of all jobs with higher priority is unknown. We show that all jobs in the higher priority queues and higher priority job arrivals can be upper and lower bounded as follows. The derivation of these asymptotically tight bounds were based on analyzing typical and atypical jobs separately and deriving bounds for each.

We describe in detail the main intuition that makes the analysis of the delay decay rate of finite-SRPT possible. For ease of understanding, we discuss the simple case of $D < M \leq 2D$. The technical proof for the general case is provided in Appendix A.5. The delay experienced by a tagged job is dependent on the volume of all jobs that are served before the tagged job leaves the system. Observe that higher priority queues and higher priority job arrivals identified through the 2DQ framework in Section 6.2 provides a clear way in which one can derive $\Pr(V^{(N)}(k) \geq m)$. In particular, the approximate analysis (approximate due to the assumption that all jobs are typical) leads to the following:

$$\Pr(V^{(N)}(k) \geq m) \approx$$

$$\begin{cases} \Pr\Big(\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0) \\ \quad + \sum_{i=1}^{k-1} iA_i^N(T_1, 1) \geq NC(m+1)\Big), & k \leq D \\ \Pr\Big(\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0) + \sum_{i=1}^{k-1} iA_i^N(T_1, 1) \\ \quad + \sum_{i=1}^{k-D-1} iA_i^N(m, T_1 + 1) \geq NC(m+1)\Big), & D < k \leq 2D. \end{cases} \tag{6.5}$$

To derive the precise expression for $\Pr(V^{(N)}(k) \geq m)$, we must improve upon Equation (6.5) in the following manner.

(i) Recall that in Section 6.2, we assumed that all jobs including the tagged job are typical. Due to this assumption, the higher priority job arrivals in Equation (6.5) are not accurate and must be corrected to take into account the atypical jobs.

(ii) The volume of higher priority jobs in $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$ must be quantified. This requires the identification of job arrivals in the past that end up in the higher priority queues at time-slot 0.

However, identifying the precise volume of all jobs that need to be served before the tagged job is difficult to determine. Our approach is to derive (asymptotically) arbitrarily close upper and lower bounds on the volume of all jobs in the higher priority queues and the volume of all higher priority job arrivals.

(i) *Bounding higher priority arrivals:* As discussed before, we need to take into account the atypical jobs in determining the higher priority job arrivals in Equation (6.5). Note that atypical jobs are generated as a result of the last job (of remaining size $\geq D$) that is served in a time-slot encountering a server with left-over capacity smaller than D. From this observation, it is clear that at most a single job can become atypical in a time-slot.

Due to this observation, the volume of higher priority arrivals for the case $k < D$ can be upper bounded by $\sum_{i=1}^{k-1} iA_i^N(m,1)$ where no job becomes atypical and lower bounded by $\sum_{i=1}^{k-1} iA_i^N(m,1) - m(M-1)$ where exactly one job becomes atypical in every time-slot and the size of the atypical job is the largest possible, i.e., $M-1$. Similarly, for the case of $D < k \leq 2D$, an upper bound is $\sum_{i=1}^{k-1} iA_i^N(m,1) + \sum_{i=1}^{k-D-1} iA_i^N(m,T_1+1)$ and a lower bound is $\sum_{i=1}^{k-1} iA_i^N(m,1) + \sum_{i=1}^{k-D-1} iA_i^N(m,T_1+1) - m(M-1)$.

(ii) *Bounding higher priority queues:* Providing bounds for the volume of all higher priority queues in Equation 6.5 is more difficult. The difficulty lies in the fact that determining the volume of jobs in the higher priority queues requires the knowledge of the past. This difficulty can be mitigated by considering typical and atypical jobs in the higher priority queues separately and derive asymptotically tight upper and lower bounds for both.

As depicted in Figure 6.2, the shaded area represents all higher priority queues, i.e., $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$. The area of higher priority queues can be separated into three distinct subareas: the thick upper-most diagonal strip (denote as B_1), the lower thick diagonal strip that is D units away from B_1 (denote at B_2), and the rest of the shaded area which we denote as C_1. Then by definition, $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + \sum_{i=1}^{k-1} Q_{i,k}(0) = B_1 + B_2 + C_1$. We make the observation that C_1 contains only atypical jobs and that all jobs in B_1, B_2, and C_1 need to be fully served before the tagged job leaves the system by a similar argument in Section 6.2. Based on this discussion, we derive asymptotically tight upper and lower bounds on $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$.

First, we consider the case where the tagged job is of size $k < D$. We consider the possible arrivals to the system that contribute to B_1, B_2, and C_1 separately, and then derive upper and lower bounds on $B_1 + B_2 + C_1$, i.e., $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$. Denote T_0 as the last time-slot before time-slot 0 when Q_k was empty, and T_{-1} as the last time before time-slot 0 when Q_{k+D} was empty. This can be generalized where

$T_i, i \leq 0$ denotes the last time-slot before 0 when Q_{k-iD} was empty.

(B_1) Note that all jobs in B_1 are jobs that arrived before time-slot 0 but have not received any service until time-slot 0. Thus, an upper bound on the volume of arrivals that contribute to B_1 is by assuming that all jobs are typical, and leads to $\sum_{i=1}^{k} iA_i^N(0, T_0 + 1)$. A lower bound on the volume of arrivals is derived by assuming that (in the worst-case) one job becomes atypical at every time-slot, and the size of the atypical job is the largest possible, i.e., $M - 1$, thus leading to the lower bound $\sum_{i=1}^{k} iA_i^N(0, T_0 + 1) + T_0(M - 1)$.

(B_2) B_2 contains jobs that had original size $\leq k + D$ and were served D units of service before time-slot 0. Thus, one can see that upper and lower bounds on the volume of job arrivals contributing to B_2 is $\sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1} + 1)$ and $\sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1} + 1) - (T_0 - T_{-1})(M - 1)$ respectively, using similar argument as for the B_1 case.

(C_1) Since all jobs in C_1 are atypical, a lower bound is 0 and an upper bound is $T_{-1}(M - 1)$.

Combining the upper and lower bounds of the contributions to B_1, B_2, and C_1 derived above with the service capacity provided during that interval, we have upper and lower bounds on $\sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$. In particular,

$$\text{Upper bound: } \sum_{i=1}^{k} iA_i^N(0, T_0 + 1) + \sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1} + 1) - T_{-1}(M - 1) - NCT_{-1},$$
$$\text{Lower bound: } \sum_{i=1}^{k} iA_i^N(0, T_0 + 1) + \sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1} + 1) + T_{-1}(M - 1) - NCT_{-1}.$$

Similar arguments hold for the case of the tagged job being size $D < k \leq 2D$. Details of the argument is omitted. The resulting upper and lower bounds are

$$\begin{aligned}\text{Upper bound:}\qquad & \sum_{i=1}^{k} iA_i^N(0, T_0+1) - T_0(M-1) - NCT_0,\\ \text{Lower bound:}\qquad & \sum_{i=1}^{k} iA_i^N(0, T_0+1) + T_0(M-1) - NCT_0.\end{aligned}$$

Combining the upper and lower bounds of the contributions to B_1, B_2, and C_1 derived above with the service capacity provided during that interval, we have upper and lower bounds on $\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) + Q_{k,k}(0)$. In particular,

$$\begin{aligned}\text{Upper bound:}\ & \sum_{i=1}^{k} iA_i^N(0, T_0+1) + \sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1}+1) - T_{-1}(M-1) - NCT_{-1},\\ \text{Lower bound:}\ & \sum_{i=1}^{k} iA_i^N(0, T_0+1) + \sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1}+1) + T_{-1}(M-1) - NCT_{-1}.\end{aligned} \tag{6.6}$$

Similar arguments hold for the case of the tagged job being size $D < k \leq 2D$. Details of the argument are omitted. The resulting upper and lower bounds are

$$\begin{aligned}\text{Upper bound:}\qquad & \sum_{i=1}^{k} iA_i^N(0, T_0+1) - T_0(M-1) - NCT_0,\\ \text{Lower bound:}\qquad & \sum_{i=1}^{k} iA_i^N(0, T_0+1) + T_0(M-1) - NCT_0.\end{aligned} \tag{6.7}$$

Combining the results, upper and lower bounds on Equation (6.5) can be derived, which leads to arbitrarily tight bounds on the virtual delay decay rate. This result can be extended to a general case which is the result in Theorem 6.3.1.

Based on the intuition, we can show that the upper bound on the decay rate, i.e., the lower bound on the probability, corresponds to the event where all jobs corresponding to the upper bound on all higher priority jobs are served before the tagged job is fully served. On

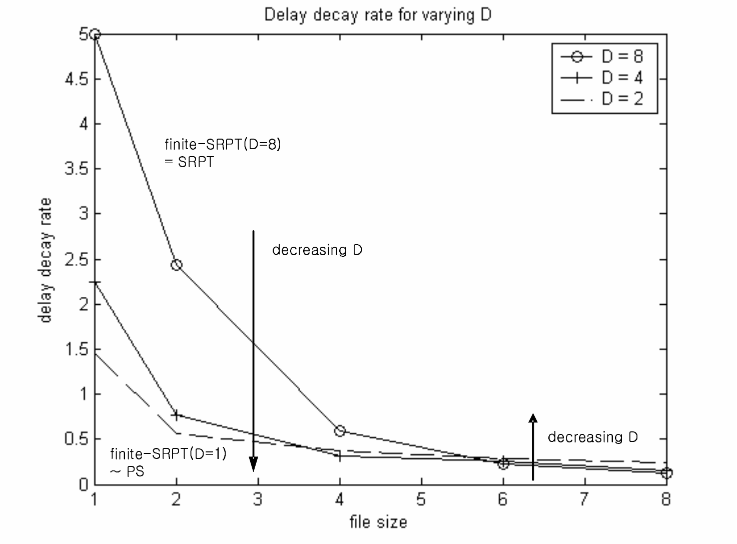


Figure 6.4: Delay decay rate of job sizes $1, 2, 4, 6, 8$, for varying $D = \{2, 4, 8\}$, exceeding the threshold $m = 5$ is depicted. X-axis is the original size of the tagged job and y-axis is the actual numerics of the delay decay rate.

the other hand, the lower bound on the decay rate, i.e., upper bound on the probability, can be derived from the event in which the lower bound on all higher priority jobs are served before the tagged job leaves the system. It is shown in [45] that as the system scales, i.e., $N \to \infty$, the difference between the volume of jobs between the upper and lower bound becomes negligible, leading to the result that the lower and the upper bound on the decay rate is tight within any $\epsilon > 0$. □

6.4 Numerical Analysis

Due to the complicated nature of the delay decay rate of finite-SRPT described in Theorem 6.3.1, it is difficult to understand its precise characteristics. Thus, in this section, we provide a numerical analysis of the delay decay rate of finite-SRPT to understand the particularly interesting question of: How does the behavior of finite-SRPT change as the maximum amount of service per time-slot (D) change?

The setup for the numerical analysis of the derived delay rate of finite-SRPT is as follows.

- Job arrives according to an on-off process (a job of size k arrives with probability p_k at each time-slot).
- Job sizes are 1, 2, 4, 6, or 8. The jobs size distribution follows the power-law with exponent 2, which is a discrete and bounded counterpart of the Pareto distribution well-known for its accuracy in describing web server requests [2].
- Capacity, C, is assumed to be 1, the load is 0.8, and the threshold value $m = 5$.

As suggested in Figure 6.3, Figure 6.4 shows that finite-SRPT is equivalent to SRPT when $D = 8$ and as D decreases to 4 and 2 finite-SRPT biases less and less toward smaller jobs. In other words, as D decreases, smaller jobs experience more delay while larger jobs experience smaller delay, i.e., the scheduling policy becomes more “fair”. The numerical results reinforces the range of the proposed finite-SRPT describe in Section 6.3.1. Thus, finite-SRPT presents a simple way in which to compromise delay performance and fairness by adjusting the maximum amount of service per time-slot. This is due to the fact that when the maximum amount of service is small, the left-over capacity after all smaller jobs have been served D units trickles down to larger jobs thus making the scheduling policy more fair.

Although not shown in this dissertation, numerical results show that the delay decay rate decrease for decreasing values of m, increasing load and for increasing M.

Chapter 7

Discrete Processor Sharing

7.1 Background and Related Work

In this section, we derive the asymptotic delay distribution of (a discretized version of) Processor-Sharing (PS), a scheduling policy known for its fairness. PS, made popular by the work of Kleinrock [22, 23], has received much attention as the idealization of time-sharing queueing models. PS models can be applied for the performance analysis of elastic traffic in integrated-service communications network and for TCP traffic in IP networks. PS is also important due to the fact that it has been shown to be the most “fair” [41].

Previous delay analysis of various scheduling policies such as FCFS [8], SJF (by applying the results of [14, 38]), SRPT (Chapter 3), and LAS (Chapter 5) relies on the observation that their priority (ordering) scheme between jobs “relatively” do not change over time. To illustrate, a job in FCFS that arrives at time t always have higher priority over jobs that come after time t and have lower priority compared to jobs that arrived before time t. In SJF, size k job always have higher priority than jobs with size $> k$ and lower priority than jobs of size $< k$. In addition, even though SRPT and LAS have time-varying priority schemes they can be reduced down to one dominating strict priority rules that do not change in time when we consider typical jobs which was shown to dominate the probability of delay (Chapter 3, Chapter 5).

However, for other scheduling policies that do not possess the time-invariant priority scheme, the derivation of the delay distribution is more difficult. Such complicated scheduling policies include policies that share the capacity in some manner such as PS, GPS [26], and Discriminatory-Processor-Sharing (DPS). The difficulty in the analysis of such scheduling policies lies in the fact that the policies themselves seem to lack any priority scheme due to their sharing nature.

The basic intuition behind the derivation of the delay characteristics of PS in the many sources large deviations regime is the observation that although PS does not seem to have a clear and coherent prioritization of jobs due to its sharing nature, PS in discrete time does indeed have clear ordering which can be taken advantage of.

In particular, we introduce the 2DQ framework with cycles, to analyze discrete PS in the many sources large deviations regime. The proposed 2DQ framework with cycles is based on the idea that in discrete time, jobs are ordered in a particular manner and that there exist time intervals (*cycles*) in which the prioritization scheme remains invariant. In particular, the state of the system (along with the priority scheme) is portrayed by the 2DQ framework and the priority schemes in 2DQs are different for each cycle while remaining invariant within a cycle.

Previous results for PS and discrete PS includes the analysis of delay distribution under the traditional assumptions of a single arrival and static capacity in $M/M/1$ [9] and in $M/G/1$ [15, 21, 49, 50]. Although the literature reports large deviation results for large buffers (i.e., study of the probability as delay tends to infinity assuming a single source) under heavy-tailed job size distribution [16, 20, 29, 50] and light-tailed job size distribution [6, 26] on the job size distribution, no previous work on the analysis of the delay distribution for many sources has been reported. This is in part due to the complexity brought on by the large number of sources. However, we show in this dissertation that the proposed 2DQ framework with cycles can simplify the analysis so that the asymptotic delay distribution is derived.

7.2 Two Dimensional Queueing Framework with cycles

In this section, we propose the 2DQ framework with cycles. The use of the 2DQ framework (described in Section 4.2) for scheduling policies such as SRPT, LAS and SMART [43] has made delay analysis in the many sources large deviations regime possible [44, 47]. However, the relatively simple structure of the 2DQ framework prevents it from being used for analysis of more complicated scheduling policies. We introduce an extension to the 2DQ framework that makes possible the analysis of more complex scheduling policies.

The analysis of simple scheduling policies such as FCFS, and SJF are possible due to the fact that they can be represented by a time-invariant priority scheme. If we focus on the dominant event for which the many sources large deviations depends on, even scheduling policies such as SRPT and SMART that seem to have changing priority schemes can be represented as a scheduling policy with time-invariant priority scheme in the 2DQ framework. For example, a typical job[1] in SRPT experiences a time-invariant priority scheme since it will be served fully once the server selects it for service. In other words, the dominant priority scheme depicted through 2DQ does not change over time for simple scheduling policies such as FCFS, SJF, SRPT and LAS. This idea will be explained in more detail in the latter part of this subsection.

In general, analyzing complicated scheduling policies that have time-varying priority schemes are difficult. However, a particular subset of these scheduling policies can be represented as time-varying priority schemes with a tractable structure. In particular, some scheduling policies can be represented as a collection of static priority schemes over continuous time intervals, which we denote as cycles. The proposed *2DQ framework with cycles takes the 2DQ framework and defines static priority schemes over continuous blocks*

---

[1] Typical jobs in SRPT are those jobs that are not preempted until they are fully served (Chapter 6, Section 6.2).

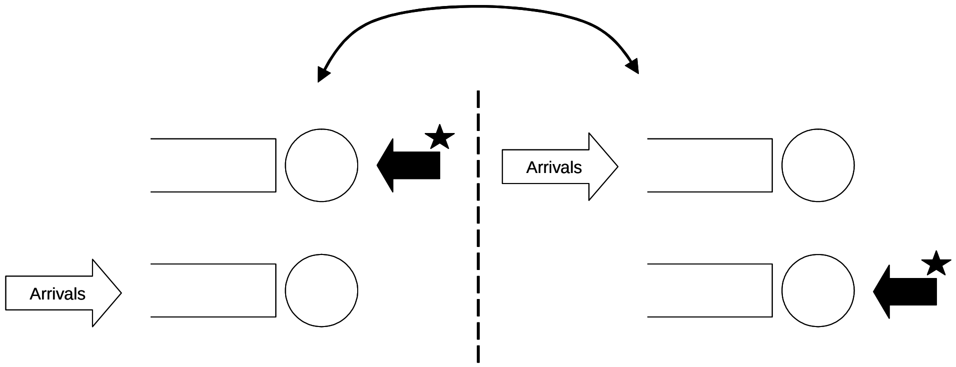


Figure 7.1: Illustration of the two queue representation of the discrete PS operation. The queue with the star arrow is the queue that is active, i.e., queue being served. A job receives a unit whence it is transferred to the other queue without the star arrow, i.e., the inactive queue. When all jobs leave the active queue, the other queue becomes active where the process repeats again.

of time slot called cycles, whereas the original 2DQ framework (without cycles) assumes a static priority scheme throughout.

7.3 Intuition

7.3.1 PS in discrete time

In the original PS, when there are $n > 0$ customers in the system, all existing customers get an equal fraction $1/n$ of the capacity. However, such PS is an idealized scheduling policy that requires the capacity to be divided infinitesimally to all jobs in a fair manner.

We consider a discretized PS scheduling policy (in the discrete time framework) that operates in the following manner. Since the smallest increment of capacity is one unit in our setup, all existing jobs found in the system at a particular time-slot should be given one unit service. After successfully distributing one unit service to all jobs, the server again sequentially serves jobs found in the buffer at that specific time-slot (time-slot when all jobs were served a unit previously). This process is repeated until the capacity available in the

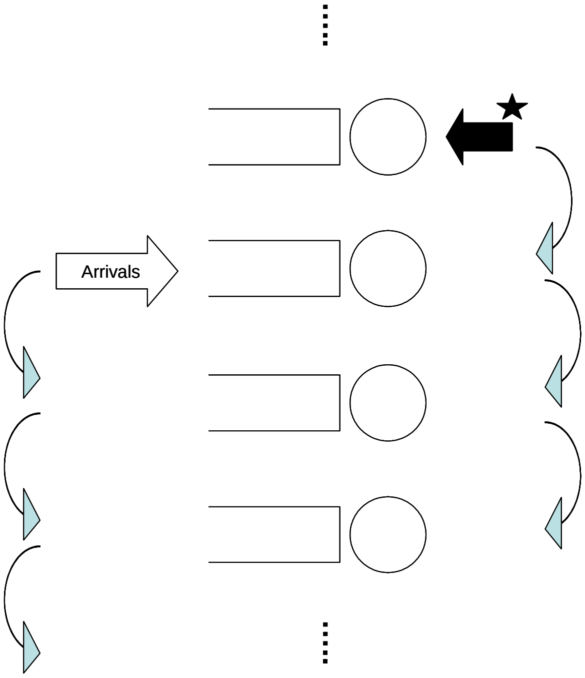


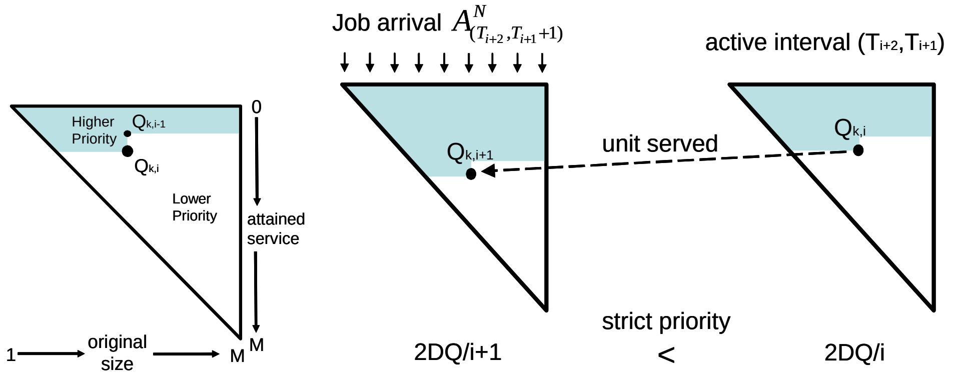
Figure 7.2: Illustration of the infinite queue representation. The argument is given that the infinite queue representation is equivalent to the two queue representation of discrete PS.

time slot runs out. At the next time slot, the process continues from where it was left off in the previous time slot.

7.3.2 Discrete PS described through 2DQ framework with cycles

In this section, we describe the operation of discrete PS using the proposed 2DQ framework with cycles. Describing the discrete PS operation using the 2DQ framework with cycles allows a more tractable description of the system state so that discrete PS can analyzed in the many sources large deviations regime.

First, we make the observation that the operation of discrete PS (described in Subsection 7.3.1) can be represented by a pair of queues as depicted in Figure 7.1. In this representation, there is an active queue that is indicated by the star arrow and an inactive



(a) Priority Scheme among jobs in 2DQ.

(b) Priority Scheme in 2DQ and position of new arrivals.

Figure 7.3: Illustration of the 2DQ with cycles representation of the discrete PS scheduling policy. The 2DQ representation follows the two queue representation and provides a tractable framework that makes the analysis of discrete PS possible.

queue (queue without the star arrow). Only jobs in the active queue are served (one unit in FCFS order) and when a job is served one unit it is transferred to the tail of the inactive queue. When the active queue becomes empty, i.e., all jobs in the active queue are served one unit, the state of the two queues are reversed, i.e., the inactive queue becomes active and vice versa as depicted in Figure 7.1. In addition, new jobs arrive to the tail of the inactive queue. We observe that this two queue representation accurately describes the operation of discrete PS, since all jobs in the system are guaranteed to receive one unit of service until a job is served again. Observe that a queue becomes active possibly in the middle of a discrete time slot, and continues to remain active until all the jobs in the queue have been served exactly one unit. This interval of time starting from a queue becoming active until it becomes empty and this inactive is called a **cycle**.

Note that the alternating transition of active queues as depicted in Figure 7.1 is equivalent to the infinite queue representation depicted in Figure 7.2, where the active queue state moves to the lower queue when the queue becomes empty. Accordingly, the inactive queue state moves to the next queue as is the case of the active queue state. Since at most

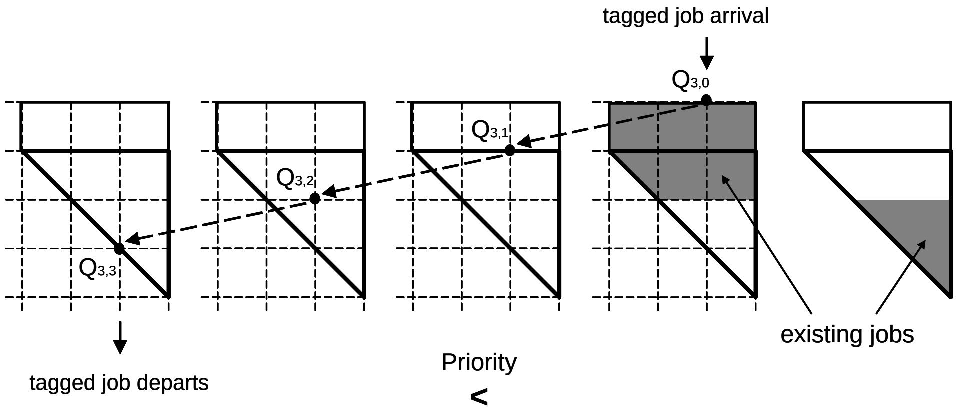


Figure 7.4: Illustration of the progress of a tagged size 3 job in the 2DQ representation. Note that the transition of active queues in the two queue representation is expressed as the sequence of multiple 2DQs with strict priority between them.

two queues (active and the inactive queues) contain jobs, all other queues in this infinite queue representation may be ignored until they become active or inactive.

Next, we connect the infinite queue representation of the operation of discrete PS to the 2DQ framework as follows. Note that the infinite queue representation of the operation of discrete PS remains identical when we replace the queues by 2DQs. A job in the active 2DQ is transferred to the appropriate queue (a queue of one more attained service) in the next 2DQ when it is served one unit as shown in Figure 7.3(b). When all jobs in the active 2DQ are served exactly one unit and becomes empty, the next 2DQ becomes active. In addition, we assume that the jobs in a 2DQ are served in the LAS order as shown in Figure 7.3(a), i.e., jobs with less attained service are served first. Note that, like SMART and LAS, this additional ordering of jobs in the individual 2DQs does not affect the asymptotic analysis of discrete PS.

Now, we explain the operation of discrete PS formally as follows. First, we make the following definition to distinguish individual 2DQs according to their respective cycles.

Definition 7.3.1. *We define **2DQ/0** as the inactive 2DQ at time slot* 0, *i.e., the 2DQ that*

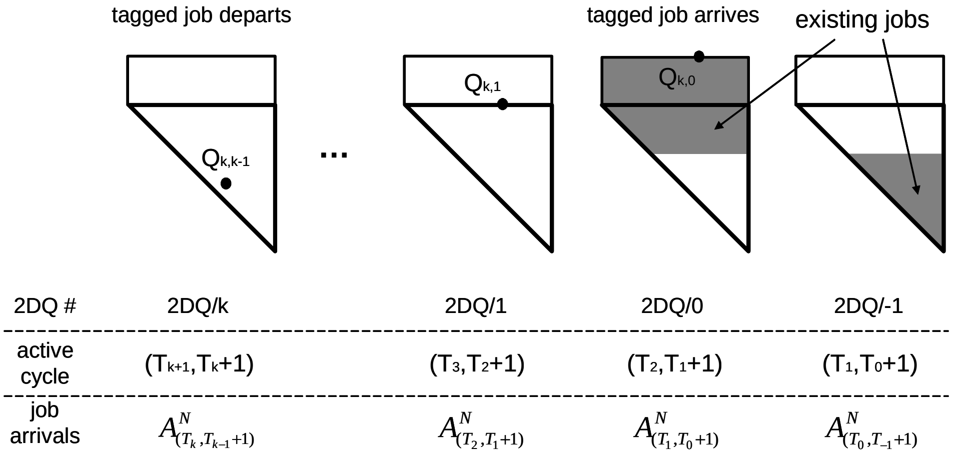


Figure 7.5: Illustration of the 2DQs representing the state of the system for a tagged job of size k after time slot 0. We denote $2DQ/0$ as the inactive 2DQ at time 0, i.e., the 2DQ that receives new arrivals at time 0. Active cycle denote the time interval that the respective 2DQ is active and the job arrivals are the actual arrivals that the particular 2DQ receives. Note that for size k tagged job only 2DQs ranging from $2DQ/0$ to $2DQ/k$ are relevant since the tagged job is fully served and leaves the system in $2DQ/k$.

accepts job arrivals at time slot 0. *Accordingly, we define* ***2DQ/T*** *as the 2DQ T cycles in the future of 2DQ/0, and* ***2DQ/-T*** *as the 2DQ T cycles in the past with respect to 2DQ/0. Thus, by definition* $2DQ/-1$ *is the 2DQ that is active at time slot* 0.

We denote $Q^r_{i,j}$ as the queue that contains all jobs having original size i that have received j unit of service and is in 2DQ/r. Thus the tagged job of size k arrives to $Q^0_{k,0}$ and then progresses to $Q^1_{k,1}, Q^2_{k,2}, \ldots, Q^{k-1}_{k,k-1}, Q^k_{k,k}$, at which point the job is fully serviced and leaves the system. We denote $Q^r_{i,j}(t)$ as the volume of $Q^r_{i,j}$ at time t, and $Q^r_i(t)$ as the volume of the queue that contains all jobs with original size i in the r'th 2DQ, i.e., $Q^r_i(t) = \sum_{j=0}^{j=i-1} Q^r_{i,j}(t)$. Similarly, denote $Q^r(t)$ as the volume of all jobs in $2DQ/r$ at time-slot t, i.e., $Q^r(t) = \sum_{i=1}^{M} \sum_{j=0}^{j=i-1} Q^r_{i,j}(t)$. A simple example shown in Figure 7.4 depicts how a tagged job of size 3 moves through the 2DQ framework with cycles until it is fully served.

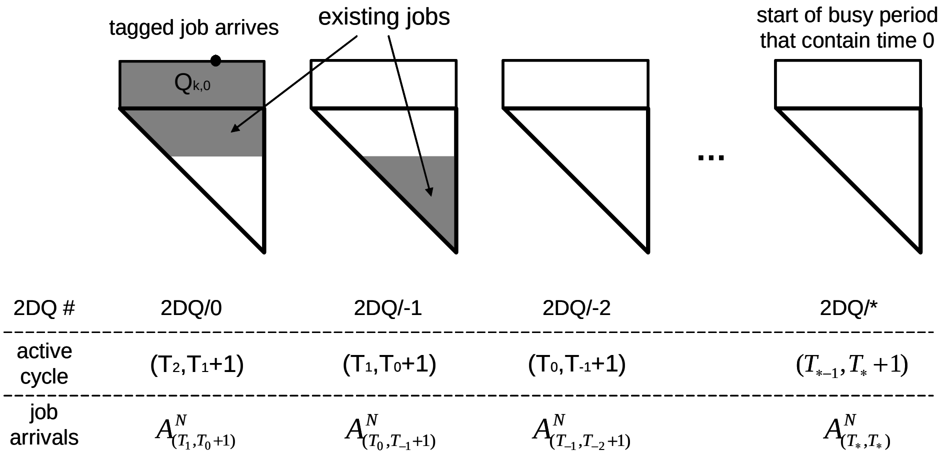


Figure 7.6: Illustration of the 2DQs representing the state of the system for a tagged job of size k before time slot 0. The 2DQ number is assigned relative to $2DQ/0$. The last 2DQ corresponds to the 2DQ that receives job arrivals at the start of the busy period, where $T_* - 1$ is the last time before time slot 0 that the system was empty.

7.4 Main Contributions

In this section, the delay decay rate of discrete PS is derived using the proposed 2DQ framework with cycles. The result of this section is the following theorem that states the delay decay rate of size k jobs under the discrete PS scheduling policy. Denote the virtual delay of discrete PS of size k job as $\hat{V}^N(k)$. First, we make the following assumption.

Assumption 7.4.1. *Let A_{high} denote the sum of all higher priority arrivals in any cycle of $(T_{i+1}, T_i + 1)$ with respect to the tagged size k job, then we assume that the corresponding rate function satisfies*

$$I^{(T_{i+1}+l,T_i+1)}_{A_{high}}\left(C(T_{i+1} - T_i + l) - v\right) < I^{(T_{i+1},T_i+1)}_{A_{high}}\left(C(T_{i+1} - T_i)\right)$$

for $v \in [v^* - \delta, v^* + \delta]$, $v^* > 0$, and $\delta > 0$ sufficiently small.

The assumption is equivalent to the decay rate being additive. Intuitively, a decay rate with the property of additive functionals implies that the occurrence of a rare event in the large deviation framework happens in a straight line. The delay decay rate of discrete PS in the many sources large deviations regime is presented in the following Theorem 7.4.1.

Theorem 7.4.1. *Under suitable straight line large deviation (Assumption 7.4.1) assumption on the arrival, the delay decay rate of size k jobs under discrete PS is*

$$I_{\hat{V}}(k,m) = \inf_{\vec{T}:\mathcal{T}} \left[\inf_{\vec{y}:\mathcal{Y}} \left\{ \sum_{j=1}^{M} I_{A_j^N(T_{i+1},T_i+1)} \left(y_j^{(T_{i+1},T_i+1)} \right) \right\} \right], \tag{7.1}$$

where condition $\mathcal{T}$ *states that* $m \geq T_k \geq T_{k-1} \geq \ldots T_1 \geq 0 \geq T_0 \ldots \geq T_{k-M+1} \geq T_{k-M} = T_*$, *and* $\mathcal{Y}$ *states that*

$$\sum_{i=k-M, i\neq 0}^{k-1} \left(\sum_{j=1}^{k-i} j y_j^{(T_{i+1},T_i+1)} + \sum_{j=k-i+1}^{M} (k-i) y_j^{(T_{i+1},T_i+1)} \right) + \sum_{j=1}^{k} j y_j^{(0,T_0+1)}$$
$$+ \sum_{j=k+1}^{M} k y_j^{(0,T_0+1)} + \sum_{j=1}^{k-1} j y_j^{(T_1,1)} + \sum_{j=k+1}^{M} (k-1) y_j^{(T_1,1)} = C(T_\star + m + 1),$$

where $y_j^{(T_{i+1},T_i+1)}, y_j^{(T_1,1)}, y_j^{(0,T_0+1)} \geq 0$ *for all* $i, j \geq 0$.

The basic idea of the proof of Theorem 7.4.1 is the following, ignoring the slight adjustment required for size k jobs. The explanation described here is only for better understanding and the actual proof considers the adjustment. Denote $[2DQ/i]^d$ and $[A^N(T_{i+1}, T_i + 1)]^d$ as d units of all jobs that arrive to $2DQ/i$ and sum of d units of all jobs in $A^N(T_{i+1}, T_i+1)$ respectively.

$$\begin{aligned} \Pr\left(\hat{V}^{(N)}(k) > m\right) &\approx \Pr\left([2DQ/k]^1 + \ldots + [2DQ/0]^{k-1} + Q^0(T_0) > NC(m+1)\right) \\ &\approx \Pr\left([2DQ/k]^1 + \ldots + [2DQ/0]^{k-1} + [2DQ/-1]^k \right. \end{aligned}$$

$$
\begin{aligned}
&\qquad + \ldots + [2DQ/k - M]^{M} > NC(T_* + m + 1)\Big) \\
= \;& \Pr\Big([A^N(T_{k+1}, T_k + 1)]^1 + \ldots + [A^N(T_1, T_0 + 1)]^{k-1} \\
&\qquad +[A^N(T_0, T_{-1} + 1)]^k + \ldots [A^N(T_{k-M+1}, T_\star + 1)]^k \\
&\qquad > NC(T_* + m + 1))
\end{aligned}
$$

Proof. Intuition(technical proof in Appendix A.6): The basic intuition behind the derivation of the delay decay rate of discrete PS are the following. Assume that the size of the tagged job is k.

(1) The number of all future 2DQs, which depict the prioritization scheme from the arrival of the tagged job until its departure, is $k + 1$. Figure 7.5 depicts all $k + 1$ future 2DQs from $2DQ/0$ to $2DQ/k$.

(2) The number of all past 2DQs, which capture the system progression from the last time the system was empty to time-slot 0, is also finite. The past 2DQs are depicted in Figure 7.6. In fact the number of relevant past 2DQs is $M - k$, i.e., $2DQ/-1$ to $2DQ/k - M$.

(3) The progress of the tagged job requires that the 2DQ in which it resides in becomes empty, i.e., all jobs in that 2DQ are served one unit. This requirement of emptying each 2DQ (serving one unit to all jobs in each 2DQ) can be translated to required service of a specific amount to all job arrivals to each 2DQs, i.e., $A^N(T_{i+1}, T_i + 1)$.

The above mentioned intuitions allows the derivation of the delay decay rate of discrete PS. First, we explain the intuition described in (1). As shown in Figure 7.5, the tagged job of size k requires $k + 1$ 2DQs to leave the system, i.e., requires k units of service. $2DQ/0$ represents its arrival and subsequent 2DQs corresponds to each unit served until $2DQ/k$ where the tagged job leaves the system once it is served another unit. Thus, only $k + 1$ future 2DQs are necessary.

We next explain the intuition described in (3) for the future 2DQs for which the same argument holds for the past 2DQs. Note that for the tagged job to leave the system, all jobs in $2DQ/0$ to $2DQ/k-1$ have to be served one unit. For $2DQ/k$ only jobs with smaller attained service and jobs with the same attained service but with smaller size need to be served one unit for the tagged job to depart. Note that this requirement is equivalent to the following statement. $k-1$ units of all jobs that arrive to $2DQ/0$, $k-2$ units of all jobs that arrive to $2DQ/1$, ..., 2 units of all jobs that arrive to $2DQ/k-1$, and 1 unit of all jobs that arrive to $2DQ/k$ are required to be served before the tagged job leaves the system, with a slight adjustment where size k job is assigned one more unit of service in all arrivals[2]. This requirement can be extended to past 2DQs so that k units of all jobs that arrive to $2DQ/-1$ are required to be served before the tagged job leaves the system and so on.

In other words, all jobs that arrive to the system in the same cycle interval will be guaranteed to be equally serviced unit by unit until the tagged job leaves the system. All jobs that arrive one cycle late will receive one less unit, and all jobs arriving one cycle early will receive one more unit of service. As described in Figure 7.5 and Figure 7.6, the relative priority of job arrivals increase and decrease as the cycle in which they arrive in moves to the future and past. This translation from service requirements to jobs in each 2DQ to service requirement of job arrivals to each 2DQ allows a simpler understanding and analysis of discrete PS.

Lastly, we discuss the specific arguments behind the intuition described in (2). Although, not as obvious as the future 2DQs, the past 2DQs can be similarly portrayed as in Figure 7.6. As shown in Figure 7.6, past 2DQs date back to the last time that the entire system was empty, i.e., T_*. The number of past 2DQs is finite due to the fact that T_* is finite, which is ensured by the stability condition. In fact, the number of all relevant past 2DQs can be simplified to be $M-k$ through the following argument. We make the observation that job arrivals to all 2DQs before $2DQ/k-M$ are required to be fully served

---

[2] This slight adjustment is due to the fact that for $2DQ/k$ only jobs with smaller attained service need to be served

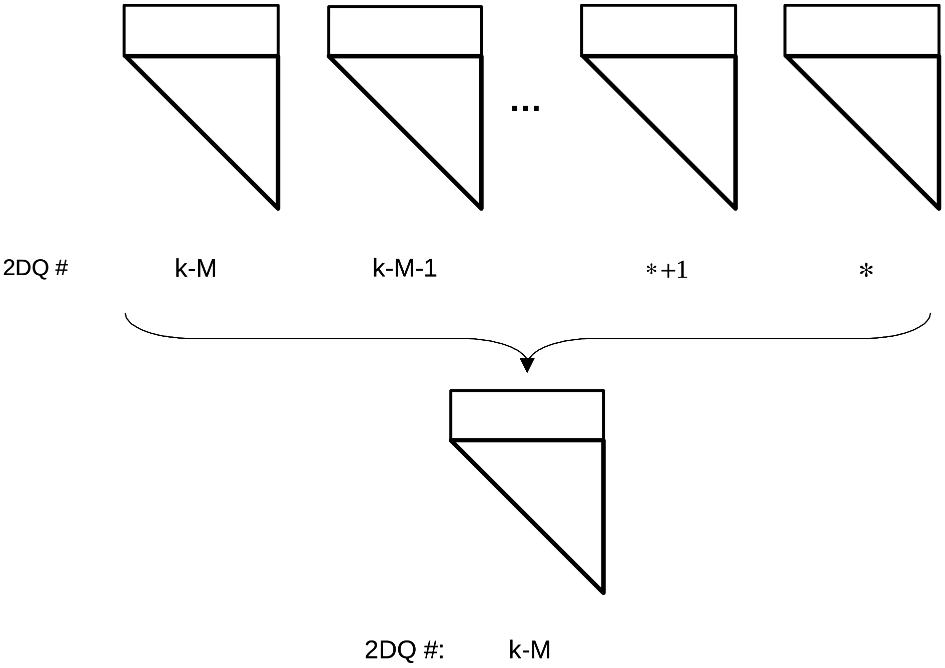


Figure 7.7: Illustration of the simplification of the past 2DQs. As depicted the unknown number of 2DQs after $2DQ/k-M$ can all be combined into a single 2DQ that represent the 2DQ with the corresponding job arrivals that need to be fully served before the tagged job leaves the system. This simplification is one of the key observations that make the analysis of discrete PS possible.

before the tagged job leaves the system, although they require different amount of units to be served before the tagged job leaves the system. This is due to the fact that job arrivals to $2DQ/k-M$ needs to be served $M-k$ units before the arrival of the tagged job (before time-slot 0) and will be served k additional units until the tagged job departs. Thus all job arrivals to 2DQs before $2DQ/k-M$ are required to be fully serviced before the tagged job leaves the system. This means that, as shown in Figure 7.7, all job arrivals to 2DQs in the following set $\{2DQ/k-M, 2DQ/k-M-1, \ldots, 2DQ/\star\}$ can be combined into a single 2DQ of $2DQ/k-M$ where the arrivals to this 2DQ are $A^N_{(T_{k-M+1},T_\star+1)}$ that are fully served before the tagged job departs. This collapse of past 2DQs results in a total of $M+1$ relevant 2DQs, i.e., $k+1$ future 2DQs and $M-k$ past 2DQs. This allows a much simpler analysis of the

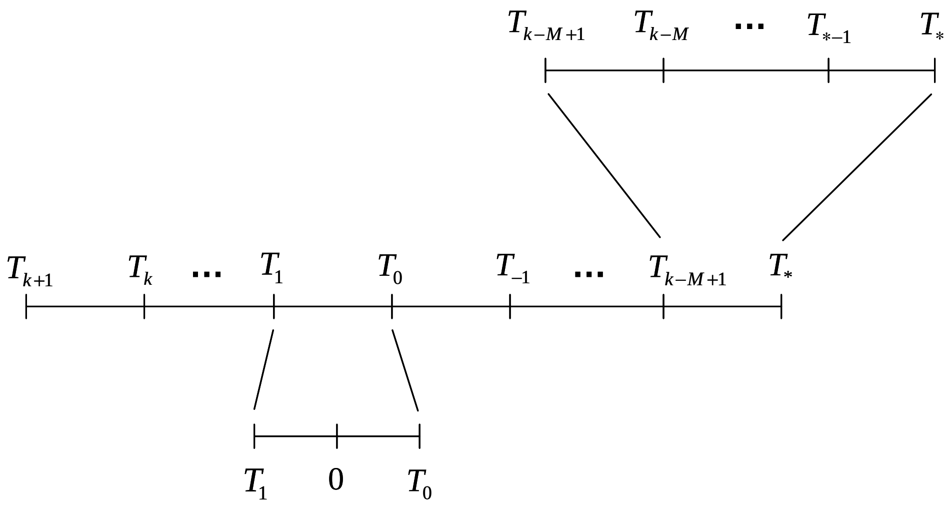
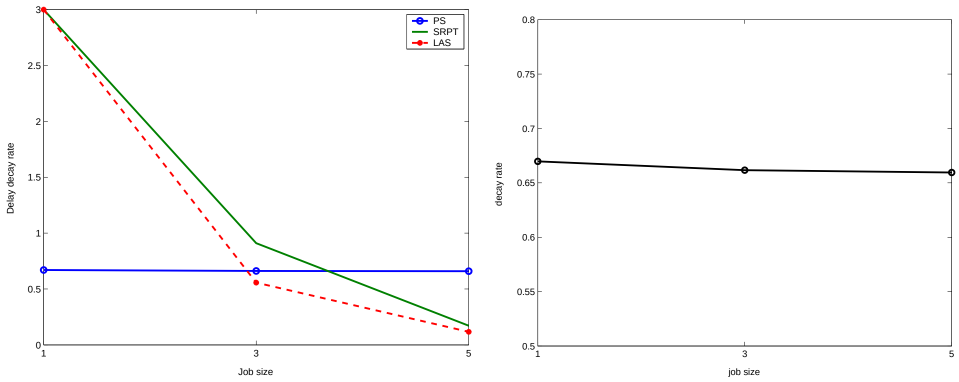


Figure 7.8: Illustration of the multiple resolution of 2DQ framework for discrete PS with respect to cycles. The figure shows the simplification of the past 2DQs and the additional resolution of $2DQ/0$ where the priority scheme before and after time 0 is different.

delay decay rate, since it much easier to track $M + 1$ 2DQs than finite but indeterminable number of 2DQs.

In addition to the fundamental intuition described above, we observe that job arrivals to $2DQ/0$, i.e., $A^N(T_1, T_0 + 1)$ must be treated differently from other cycle due to the following facts: (1) the tagged job arrives in this cycle at time 0, i.e., $T_1 \leq 0 \leq T_0 + 1$, and that (2) due to the FCFS ordering between jobs with the same original and remaining size, size k jobs that arrive before the tagged job receive an additional unit compared to the jobs that arrive after the tagged job. Thus, as depicted in Figure 7.8, the cycle $(T_1, T_0 + 1)$ should be separated into cycles of $(T_1, 1)$ and $(0, T_0 + 1)$ where the only difference is that an additional unit is served for size k jobs of cycle $(T_1, 1)$ compared to $(0, T_0 + 1)$. This finer resolution of the cycle $(T_1, T_0 + 1)$ and coarser resolution of cycles $(T_{k-M+1}, T_{k-M} + 1), \ldots, (T_{\star+1}, T_\star + 1)$ is depicted in Figure 7.8.

Lastly, the last cycle $((T_{k+1}, T_k))$ must also be treated differently from the others since not all jobs should be served. Only the jobs in higher priority queues (queues with



(a) Delay decay rate of discrete PS compared to SRPT and LAS.

(b) Delay decay rate of discrete PS.

Figure 7.9: Plot of the delay decay rate of discrete PS under the exponential distribution with $\rho = 0.8$, $m = 5$, and $C = 1$ for jobs sizes $1, 3$, and 5. The asymptotic decay rate of discrete PS shows that even the exponential decay rate of discrete PS is dependent on the job size where the decay rate decreases for larger jobs, i.e., slightly favors small jobs. However, compared to SRPT and LAS the discrete PS policy does not favor small jobs as much. The delay decay rate of SRPT and LAS for size 1 jobs are actually infinite, but depicted as finite to able to compare the delay decay rate of other job sizes.

attained service $\leq k-1$ and queues of attained service $= k$ and original size $\leq k$ need to be served a unit service instead of all the jobs in the 2DQ as the other cycles. $\square$

7.5 Numerical Analysis

We consider an On-Off source in which at every time-slot a job of size k arrives to the system with probability p_k . The job sizes are either 1, 3, or 5. The jobs size distribution follows an exponential distribution. The per flow capacity is $C = 1$ and the load is set to be 0.8. Based on this setup, we derive the numerics of the delay decay rate of discrete PS for each job sizes when the threshold value is $m = 5$.

The numerical analysis for the above mentioned scenario is depicted in Figure 7.9. Figure 7.9(a) shows that compared to SRPT and LAS, discrete PS does not seem to favor small jobs. However, Figure 7.9(b) shows that discrete PS grants slight preference to smaller sized jobs but not overly much. This is an expected result since discrete PS exhibit different mean delay that is proportional to the job size, i.e., discrete PS has equal mean slowdown across job size, which should be reflected in the delay decay rate.

Chapter 8

Concluding Remarks

Scheduling is a key consideration for efficient resource allocation that can affect the performance of the computer systems especially in bottlenecks. In literature, much of the analysis of scheduling policies have depended on traditional queueing models that captures the properties of small to medium sized systems. However, extension of these models to large scale systems mostly entails approximations or omissions of important characteristics of large scale systems. Understanding scheduling under a model specifically tailored to capture the properties of large scale systems is important to precisely understand how and why scheduling behaves as they do in current computer systems. In this dissertation, we consider the many sources regime for the analysis and modeling of scheduling in web servers.

In Chapter 3, we derive the delay decay rate of the Shortest-Remaining-Processing-Time (SRPT) scheduling policy under the many sources regime. Although SRPT is the optimal scheduling policy in terms of mean delay, it has been believed that in the process of optimizing the mean delay, fairness among jobs of different sizes might suffer, i.e., “starvation” of larger jobs. Our results on the delay decay rate of SRPT shows that compared to the prevalent First-Come-First-Serve (FCFS) the unfairness is quite small. In fact, the decay rate difference between SRPT and FCFS decreases in proportion to the inverse of the job size when the job size distribution is of heavy tail.

The results of Chapter 3 point to the fact that SRPT is a viable policy for web servers in large scale systems. However, there are various implementation constraints for SRPT which are addressed in Chapter 4 to Chapter 6. Chapter 4 addresses the fact that actual implementations are variants of the ideal SRPT. We derive the asymptotic delay tail probabilities across job sizes of a class of scheduling policies called SMART that include SRPT and its variants. The results show that all scheduling policies that obey certain properties that characterize SMART have the same asymptotic delay characteristics as the mean delay optimal SRPT.

Chapter 5 addresses the implementation constraint of SRPT for the case where the remaining processing time information is not available at the web servers. It has been shown in the literature and in practice that the Least-Attained-Service (LAS) scheduling policy is a good substitute for SRPT when the job size distribution is heavy tailed. We study its delay characteristics in the many sources large deviations regime and show that the penalty in using the attained service information (LAS) rather than the remaining processing time (SRPT) is prevalent in that LAS is always worse than SRPT across all job sizes.

Actual implementations of any scheduling policy must take into account the end user bandwidth (BW) constraints that exist in web servers. However, the ideal SRPT or SMART assume that a job selected for service receives the full attention of the server until it is fully served or is preempted. In Chapter 6, we consider the end user BW constraint on SRPT (finite-SRPT) and derive its delay decay rate. The results show that as the BW constraint increase finite-SRPT approaches the ideal SRPT and as the constraint decrease finite-SRPT becomes more fair, i.e., similar to the fair Processor-Sharing (PS) scheduling policy.

Finite-SRPT brings us to the question of the characteristics of the “fair” scheduling policy, PS, when the web server is accessed by a large number of flows. We investigate the asymptotic delay characteristics of a discrete time PS in the many sources regime in Chapter 7. The delay decay rate of PS shows that the exponential decay rate of a fair policy is not equal for all job sizes, rather the decay rate decreases as the job size increases.

An interesting and practical issue not discussed in this dissertation is the delay performance of scheduling policies accessed by a large number of sources in multi server setting or server farm setting. It would be of interest to investigate the asymptotic delay characteristics of various scheduling policies in such settings and to provide practitioners good directions on how to dispatch jobs to different servers.

Appendix A

Proof of Theorems

A.1 Proof of Theorem 3.2.1

First, we derive the lower bound, i.e., $I^{(k)}_{V_{C-\epsilon}}(m) \leq I^{(k)}_{\overline{V}}(m)$. We denote by $\overline{B}^{N}_{k}(a,b)$, the volume of potential service[1] that jobs in Q_k can receive in an interval (a,b) under SRPT. Observe that if the virtual delay exceeds m, then we have that the total queue length[2] at time zero (i.e., $Q_k(0)$) is not served by time m. In other words, $\{V^{(N)}(k) > m\} \subset \{Q_k(0) > \overline{B}^{N}_{k}(1,m)\}$, and consequently

$$\Pr(V^{(N)}(k) > m) \leq \Pr(Q_k(0) > \overline{B}^{N}_{k}(1,m)). \tag{A.1}$$

From Loynes' formula, we have

$$\Pr(Q_k(0) > \overline{B}^{N}_{k}(-T,m)) = \Pr(\sup_{T \geq 0}[A^{N}_{k}(-T,0) + \overline{S}^{N}_{k}(-T,0) - \overline{B}^{N}_{k}(-T,m)] \geq 0), \tag{A.2}$$

---

[1] Potential service corresponds to the maximum amount of service that can be received if the corresponding queue is not empty.

[2] The unit of the queue length is the volume of data. Thus, for Q_k, $Q_k(0)$ denotes k times the number of size k jobs in the queue.

where $\overline{S}_k^N(-T,0)$ is the volume of arrivals to Q_k due to partially served jobs arriving from lower priority queues, $Q_{>k}$, in the interval $(-T,0)$. Let $-T^*$ be the first time before time 0 such that $Q_k(-T^*-1)=0$. Without loss of generality, we can show that $Q_l(-T^*-1)=0$, for all $l \leq k$ (the proof is provided in Theorem 4.1 of [38] in the context of priority queues). Hence, we have

$$\begin{aligned} & \Pr(\sup_{T\geq 0}[A_k^N(-T,0)+\overline{S}_k^N(-T,0)-\overline{B}_k^N(-T,m)] \geq 0) \\ = \; & \Pr(A_k^N(-T^*,0)+\overline{S}_k^N(-T^*,0)-\overline{B}_k^N(-T^*,m) \geq 0). \end{aligned} \tag{A.3}$$

Since all $Q_{\leq k}$ are empty at time $-T^*-1$, the potential service available to Q_k in the interval $(-T^*,m)$ is lower bounded by the residual service after all external arrivals to $Q_{\leq k}$ and internal arrivals to $Q_{\leq k}$ generated by partially served jobs are served, i.e.,

$$\overline{B}_k^N(-T^*,m) \geq NC(T^*+m+1) - \sum_{i=1}^{k-1} A_i^N(-T^*,m) - (T^*+m+1)(k-1). \tag{A.4}$$

Note that in Equation (A.4), the term $(T^*+m+1)(k-1)$ accounts for the worst case scenario where at every time slot in $(-T^*,m)$, a partially served job arrives at Q_{k-1} from queues with higher priority than k. This observation follow from the fact that at most only one additional job can be partially served in a time slot.

From Equation (A.1), Equation (A.2), Equation (A.3), Equation (A.4), and the fact that $\overline{S}_k^N(-T^*,0) \leq k(T^*+1)$, we have

$$\begin{aligned} \Pr(V^{(N)}(k) > m) \; & \leq \; \Pr[A_k^N(-T^*,0)+\overline{S}_k^N(-T^*,0)-\overline{B}_k^N(-T^*,m) > 0] \\ & \leq \; \Pr[A_k^N(-T^*,0)+\sum_{i=1}^{k-1} A_i^N(-T^*,m) - NC(T^*+m+1) \\ & \quad +(T^*+1)k+(T^*+m+1)(k-1) > 0] \end{aligned}$$

$$
\begin{aligned}
&\leq \Pr[A_k^N(-T^*,0) + \sum_{i=1}^{k-1} A_i^N(-T^*,m) - NC(T^*+m+1) \\
&\quad +2(T^*+m+1)k > 0] \\
&\leq \Pr[A_k^N(-T^*,0) + \sum_{i=1}^{k-1} A_i^N(-T^*,m) - N(C-\frac{2k}{N})(T^*+m+1) > 0] \\
&\leq \Pr[\bigcup_{T\geq 0}(A_k^N(-T,0) + \sum_{i=1}^{k-1} A_i^N(-T,m) - N(C-\frac{2k}{N})(T+m+1) > 0)] \\
&\leq \sum_{T\geq 0}\Pr[A_k^N(-T,0) + \sum_{i=1}^{k-1} A_i^N(-T,m) - N(C-\frac{2k}{N})(T+m+1) > 0].
\end{aligned}
$$

Fix any $\epsilon > 0$. Observe that for N large enough, we have $(C - \frac{2k}{N}) > (C - \epsilon)$. Hence,

$$
\Pr(V^{(N)}(k) > m) \leq \sum_{T\geq 0}\Pr[A_k^N(-T,0) + \sum_{i=1}^{k-1} A_i^N(-T,m) - N(C-\epsilon)(T+m+1) > 0]. \quad \text{(A.5)}
$$

Note that Equation (A.5) is the same expression for the rate function of size k jobs in priority queues with capacity $C - \epsilon$. (This uses Assumption 3.2.1 as in [13, 24].) Using similar techniques as in [13, 24], it follows that the lower bound of the rate function of $\Pr(Q_k(0) > \overline{B}_k^N(1,m))$ is $I^{(k)}_{V_{C-\epsilon}}(m)$.

Next, we derive the upperbound, i.e., $I^{(k)}_{\overline{V}}(m) \leq I^{(k)}_{V_{C+\epsilon}}(m)$. Since a lower bound on the probability is an upper bound of the rate function, we concentrate on finding a lower bound on $\Pr(V^{(N)}(k) > m)$. We do so by constructing a priority queueing based system which lower bounds the delay experienced in the SRPT scheduler.

As a basis for comparison, we define PRI-0 to be a priority queueing system with capacity NC. This system consists of M queues, with size k jobs arriving to Q_k. Partially served jobs in this system continue to reside in the same queue, i.e., no switching of jobs occur. Next, we consider a priority queueing system PRI-1, with capacity NC, where *all partially served jobs completely leave the system*, instead of residing in the same queue (PRI-0) or switching to a higher priority queue (SRPT). By construction, this system has fewer arrivals to each queue, and at least as many departures from each queue as compared to the SRPT scheduler. Thus, PRI-1 provides a lower bound on the delay experienced by a job compared to the SRPT scheduler.

Next, we fix an $\epsilon > 0$, and consider PRI-2, a priority queueing system with capacity $N(C + \epsilon)$. Thus, the operation of PRI-2 is identical to that of PRI-0, but with additional service capacity of $N\epsilon$. First, we compare PRI-1 and PRI-0. In any time slot, at most only one job can be partially served. This implies that the maximum additional potential service that Q_k can receive in PRI-1 compared to PRI-0 is $(k - 1)$. On the other hand, for any $N \geq M/\epsilon$, the system PRI-2 will provide an additional service of $N\epsilon \geq M \geq k$, compared to PRI-0. Thus, PRI-2 provides more potential service to Q_k compared to PRI-1. Further, note that PRI-2 has the same number of external arrivals as PRI-1. Thus, PRI-2 provides a lower bound on the virtual delay of a job compared to PRI-1, and consequently a lower bound to that of the SRPT scheduler.

We now describe the above argument in greater detail. Consider the case where there are two queues: size 1 and size M. Let the queues for PRI-1 be $Q_1^{(1)}(t)$ and $Q_M^{(1)}(t)$, and the queues for PRI-2 be $Q_1^{(2)}(t)$ and $Q_M^{(2)}(t)$. As described above, the arrival processes to PRI-1 and PRI-2 systems are the same. However, the potential service for $Q_M^{(1)}(t)$ and $Q_M^{(2)}(t)$ are different. For PRI-1, we have that the potential service at time t to Q_M is upper bounded by the sum of $(NC - Q_1^{(1)}(t))$ and (possibly) partially served size M jobs. Thus, the upper bound on the potential service for $Q_M^{(1)}(t)$ is $(NC - Q_1^{(1)}(t) + M)$. On the other hand, $Q_M^{(2)}(t)$ has potential service of $(NC - Q_1^{(2)}(t) + N\epsilon)$. Further, we have that $Q_1^{(2)}(t) \leq Q_1^{(1)}(t)$. This is due to the following three facts: (i) the external arrivals to $Q_1^{(2)}(t)$ and $Q_1^{(1)}(t)$ are the same, (ii) jobs of size 1 are fully served (i.e., there is no partially served size 1 job), and (iii) PRI-2 has larger capacity than PRI-1. Combining the fact that $Q_1^{(2)}(t) \leq Q_1^{(1)}(t)$, and that $N\epsilon \geq M \geq k$, we have that the potential service provided by PRI-1 is no more than PRI-2. This argument can be directly extended to case of multiple queues. Thus, the delay experienced by a job in a priority queueing system with capacity $N(C + \epsilon)$ is a lower bound on the delay of jobs in SRPT. Thus, we have that $I_{\overline{V}}^{(k)}(m) \leq I_{V_{C+\epsilon}}^{(k)}(m)$.

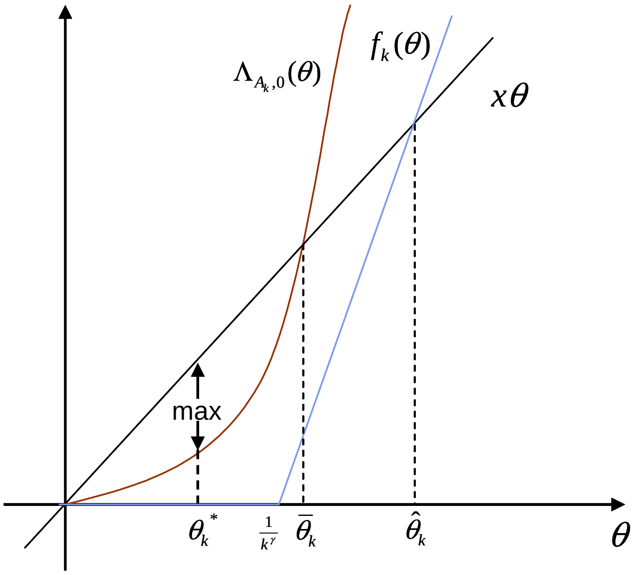


Figure A.1: Illustration of the bounds of θ^* that are used to prove Theorem A.2.1. As shown in the figure, we construct $f_k(\theta)$ that lower bounds $\Lambda_{A_k,0}(\theta)$ for all θ.

A.2 Proof of Theorem 3.2.2

We first derive an upper bound on the delay rate function of SRPT and FIFO. The upper bound on the delay rate function of size k jobs for SRPT is given by selecting specific values in the infimizing set, i.e., $I^{(k)}_{\overline{V}}(m) \leq \inf_{T\geq 0}[I_{A_k,T}((C-\alpha)(T+m+1))] \leq I_{A_k,0}((C-\alpha)(m+1))$ (since $I_{A_i,0}(E[A_i(0,0)]) = 0$). Using the same technique, an upper bound of the delay rate function for FIFO is $I_{\hat{V}}(m) \leq I_{A_k,0}(C(m+1)-\alpha_M)$.

From the upper bounds derived above and the fact that the rate functions are non-negative, it follows that

$$|I^{(k)}_{\overline{V}}(m) - I_{\hat{V}}(m)| \leq \max\{I_{A_k,0}((C-\alpha)(m+1)), I_{A_k,0}(C(m+1)-\alpha_M)\}. \tag{A.6}$$

Based on Assumption 3.2.3 on the marginal probabilities of the arrival process, a more revealing upper bound of $I_{A_k,0}(x)$ using Equation (A.6) is derived in the following theorem.

Theorem A.2.1. *Fix any* $0 < \gamma < (1 - \eta)$*, and* $x > E[A_k(0,0)]$*. Then there exists* $\bar{K}(x)$ *such that*

$$I_{A_k,0}(x) \leq \frac{x(x+1)}{k^\gamma} \qquad \forall k \geq \bar{K}(x).$$

Proof. First, we make some basic observations for the proof. From the i.i.d. assumption across flows, we have $\Lambda_{A_k,0}(\theta) = \log E(e^{\theta A_k(0,0)})$. We denote θ_k^* as the θ that satisfies $I_{A_k,0}(x) = \sup_{\theta \in \mathbb{R}}(x\theta - \Lambda_{A_k,0}(\theta))$. Further, for $A_k(0,0)$ satisfying Assumption 3.2.3, $\Lambda_{A_k,0}(\theta)$ is convex, non–negative, and is increasing for $\theta \geq 0$ with $\Lambda_{A_k,0}(0) = 0$ (see Figure A.1). Further, since $x > E(A_k(0,0))$, we can restrict the supremizing set of θ to $\{\theta \geq 0\}$.

We define $f_k(\theta)$ as follows. For any $0 < \gamma < (1 - \eta)$ and for all $k > \bar{K}(x) = \max(K_\eta, 2(x+1))$,

$$f_k(\theta) = \begin{cases} 0 & 0 \leq \theta \leq 1/k^\gamma \\ (x+1)(\theta - \frac{1}{k^\gamma}) & \theta > 1/k^\gamma. \end{cases}$$

The function $f_k(\theta)$ is constructed such that for all $k > \bar{K}(x)$ it is a lower-bound of $\Lambda_{A_k,0}(\theta)$ (see Figure A.1). To show this, first observe that for all $\theta \leq 1/k^\gamma$, we have $0 = f_k(\theta) \leq \Lambda_{A_k,0}(\theta)$. Further, for $\theta = 1/k^\gamma$, we have

$$\begin{aligned} \left[\frac{d\Lambda_{A_k,0}(\theta)}{d\theta}\right]_{\theta=\frac{1}{k^\gamma}} &= \frac{kq(k)e^{k^{1-\gamma}}}{p(k) + q(k)e^{k^{1-\gamma}}} \geq \frac{kq(k)e^{k^{1-\gamma}}}{1 + q(k)e^{k^{1-\gamma}}} = k\frac{1}{1 + \frac{1}{q(k)e^{k^{1-\gamma}}}} \\ &\geq k\frac{1}{1 + \frac{1}{A_\eta}} \geq \frac{k}{2} \geq (x+1) \quad \forall k > \bar{K}(x). \end{aligned}$$

Since $\Lambda_{A_k,0}(\theta)$ is convex, it follows that $\frac{d\Lambda_{A_k,0}(\theta)}{d\theta} \geq x+1$ for all $\theta \geq 1/k^{\gamma}$ and $k > \overline{K}(x)$. Further, as $f_k(1/k^{\gamma}) \leq \Lambda_{A_k,0}(1/k^{\gamma})$, it follows that $f_k(\theta) \leq \Lambda_{A_k,0}(\theta)$ for all $\theta \geq 0$ and $k > \overline{K}(x)$.

We define $\overline{\theta}_k$ as the θ that satisfies $x\theta = \Lambda_{A_k,0}(\theta)$, and $\hat{\theta}_k$ as the θ that satisfies $x\theta = f_k(\theta)$. Note that $\overline{\theta}_k$ and $\hat{\theta}_k$ depends on k. Since $x\theta$ is affine and $\Lambda_{A_k,0}(\theta)$ is convex, we have $\theta_k^* \leq \overline{\theta}_k$. Furthermore, as shown above, $f_k(\theta) \leq \Lambda_{A_k,0}(\theta)$ for all $\theta \geq 0$ (see Figure A.1), thus we have $\overline{\theta}_k \leq \hat{\theta}_k$. Thus, an upper bound on $I_{A_k,0}(x)$ is $x\hat{\theta}_k$, since $I_{A_k,0}(x) = x\theta_k^* - \Lambda_{A_k,0}(\theta_k^*) \leq x\theta_k^* \leq x\overline{\theta}_k \leq x\hat{\theta}_k$. Computing $\hat{\theta}_k$, we have $\hat{\theta}_k = \frac{x+1}{k^{\gamma}}$. Consequently

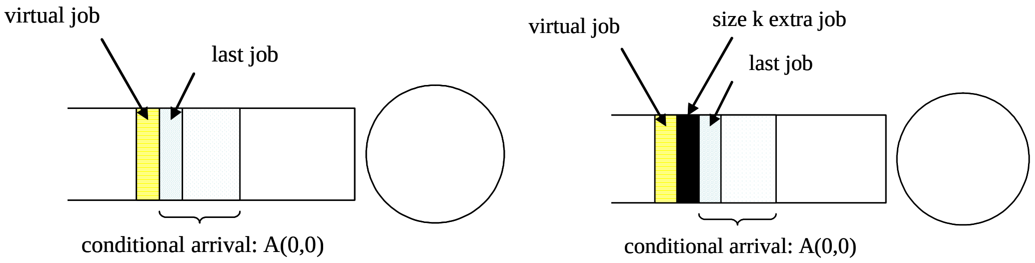
$$I_{A_k}(x) \leq \frac{x(x+1)}{k^{\gamma}} \qquad \forall k > \overline{K}(x).$$

$\square$

As a corollary of Theorem A.2.1, the upper bound on the difference of the delay rate function for size k jobs in SRPT and FIFO is derived, i.e., Theorem 3.2.2.

Proof. Combining Theorem A.2.1 and upper bounds of delay rate function for size k jobs under SRPT and FIFO derived previously, we have the following upper bound on the rate functions. Denote $c_1 = (C-\alpha)(m+1)$, $c_2 = C(m+1) - \alpha_M$ and note that $c_1, c_2 > \beta$, where $\beta = E[A_k(0,0)]$. For any fixed $0 < \gamma < 1-\eta$ there exists $\overline{K}_1(m)$ and $\overline{K}_2(m)$ such that $I_{\overline{V}}^{(k)}(m) \leq \frac{c_1(c_1+1)}{k^{\gamma}}$ for all $k \geq \overline{K}_1(m)$, and $I_{\hat{V}}(m) \leq \frac{c_2(c_2+1)}{k^{\gamma}}$ for all $k \geq \overline{K}_2(m)$. Applying Equation (A.6), and the fact that the rate function is non–negative, we have that the upper bound on the difference between the rate functions for k sized jobs of delay asymptote for SRPT and FIFO is

$$|I_{\overline{V}}^{(k)}(m) - I_{\hat{V}}(m)| \leq \max\{\frac{c_1(c_1+1)}{k^{\gamma}}, \frac{c_2(c_2+1)}{k^{\gamma}}\}$$



(a) Queue state for the lower bound. (b) Queue state for the upper bound.

Figure A.2: Illustration of the lower and upper bound of the actual delay with respect to the virtual delay. The lower bound shows that the actual job can leave the system only if the virtual job leaves the system, i.e., reaches the head of the queue. The upper bound shows that if the virtual job behind the additional size k job reaches the head of the queue then the actual job is guaranteed to be fully served.

for all $k \geq \max\{\overline{K}_1(m), \overline{K}_2(m)\} = \hat{K}(m)$, which proves the theorem. □

A.3 Proof of Theorem 4.4.1

We now derive the actual delay decay rate of SMART in the many sources regime. We prove Theorem 4.4.1 by first considering the virtual delay, $\Pr(\overline{V}^{(N)}(k) > m)$, for size k jobs then deriving the actual delay, $\Pr(\overline{W}^{(N)}(k) > m)$.

Define $I_{\overline{W}}(k, m)$ as the decay rate of the actual delay of a size k job under SMART with total service rate NC. Let $I_{\overline{V}_\mu}(k, m)$ denote the decay rate of the virtual delay of a size k job under SMART with total service rate $N\mu$. For bounding purposes, we consider the virtual delay of SMART where an additional size k job is inserted before the virtual job, and denote it as $I_{\tilde{V}_\mu}(k, m)$.

Lemma A.3.1. *For any $k \in \mathcal{M}$, under any scheduling policy in SMART*

$$I_{\tilde{V}_C}(k, m) \leq I_{\overline{W}}(k, m) \leq I_{\overline{V}_C}(k, m). \tag{A.7}$$

Proof. First, we derive the upper bound by showing

$$\Pr\left(\overline{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \le \Pr\left(\overline{W}^{(N)}(k) > m\right). \tag{A.8}$$

Note that a virtual job (with size 0) need only to arrive at the front of the queue to be fully serviced. Thus, Equation (A.8) follows from the observation that, if a fictitious job did not leave the system (i.e. arrive at the head of the queue) before time m, then the actual job did not leave the queue. That is, the actual job did not even receive one unit of service. Thus, the actual job is guaranteed to have not left the system by time m. This queue state is depicted in Figure A.2(a), where the last job corresponds to the actual job. Thus, we have

$$-\frac{1}{N}\log\Pr\left(\overline{W}^{(N)}(k) > m\right) \le -\frac{1}{N}\log\Pr\left(\overline{V}^{(N)}(k) > m | A^N(0,0) > 0\right). \tag{A.9}$$

As $N \to \infty$, Equation (A.9) can be further upper bounded by $I_{\overline{V}_C}(k,m)$ using similar techniques to those in [38]. Finally,

$$-\frac{1}{N}\log\Pr\left(\overline{W}^{(N)}(k) > m\right) \underset{N\to\infty}{\longrightarrow} I_{\overline{W}}(k,m)$$

gives the upper bound.

To prove the lower bound, we add an extra size k job in front of the virtual queue and consider the virtual delay, $\tilde{V}^{(N)}(k)$. The queue state is depicted in Figure A.2(b). We prove the following inequality using a contra-positive argument.

$$\Pr(\overline{W}^{(N)}(k) > m) \quad \le \quad \Pr\left(\tilde{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \tag{A.10}$$

The event $\left\{\tilde{V}^{(N)}(k) \le m | A^N(0,0) > 0\right\}$ is equivalent to the statement that the virtual job leaves the system before time m. Note that for a virtual job to leave the system, all that is required is for the virtual job to reach the head of the queue. The virtual job reaches the head of the queue when the extra job leaves the queue, i.e., the extra job gets one unit of service. The key observation is that, the extra job need not be fully serviced, only partially serviced. However, for the extra job to be even partially serviced, the last job of the batch arrival must leave the system completely. This is due to the secondary priority scheme introduced in the two dimensional queueing framework: jobs with the same original size but with smaller remaining sizes have higher priority. This last job in the front of the virtual job is the job that represents the actual delay. Thus, we have the following.

$$\left\{\tilde{V}^{(N)}(k) \leq m | A^N(0,0) > 0\right\} \Rightarrow \left\{\overline{W}^{(N)}(k) \leq m\right\},$$

where $A \Rightarrow B$ denotes A implies B. This proves Equation (A.10) by a contra-positive argument. Thus, we have

$$-\frac{1}{N} \log \Pr\left(\tilde{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \leq -\frac{1}{N} \log \Pr\left(\overline{W}^{(N)}(k) > m\right).$$

As $N \to \infty$, above equation can be lower bounded by $I_{\tilde{V}_C}(k,m)$ using similar arguments to [38]. Finally, noting the definition of $I_{\overline{W}}(k,m)$, completes the proof. □

We are now ready to prove Theorem 4.4.1. Define $\overline{B}^{N}_{(k,k)}(a,b)$ as the volume of potential service that jobs in $Q_{k,k}$ can receive in interval (a,b) under SMART. Potential service corresponds to the maximum amount of service that can be received if the corresponding queue is never empty. Note that $Q_{k,k}$ does not include original size k jobs that have received partial service.

Proof. (of Theorem 4.4.1) First, we derive the lower bound on the decay rate in Equation (4.3) by finding an upper bound of $\Pr\left(\tilde{V}^{(N)}(k) > m\right)$ and using Lemma A.3.1.

Let us consider the virtual delay in SMART with the extra job, i.e., $\Pr\left(\tilde{V}^{(N)}(k) > m\right)$. Observe that if the virtual delay with the extra size k job exceeds m, then we have that the queue length at time zero (i.e. $Q_{k,k}(0)$) and the extra job is not served by time m. In other words, $\left\{\tilde{V}^{(N)}(k) > m\right\} \Rightarrow \left\{Q_{k,k}(0) + k > \overline{B}^{N}_{(k,k)}(1, m)\right\}$, which results in

$$\Pr\left(\tilde{V}^{(N)}(k) > m\right) \leq \Pr\left(Q_{k,k}(0) + k > \overline{B}^{N}_{(k,k)}(1, m)\right). \tag{A.11}$$

From Loynes' formula, we have

$$\begin{aligned}\Pr\left(Q_{k,k}(0) + k > \overline{B}^{N}_{(k,k)}(1, m)\right) &= \Pr\left(\sup_{T\geq 0}\left[kA^{N}_{k}(-T, 0) + k - \overline{B}^{N}_{(k,k)}(-T, m)\right] \geq 0\right) \\ &= \Pr\left(kA^{N}_{k}(-T^{*}, 0) - \overline{B}^{N}_{(k,k)}(-T^{*}, m) + k \geq 0\right).\end{aligned} \tag{A.12}$$

Note that in addition to the volume of $Q_{k,k}$ ($Q_{k,k}(0)$), k is added in deriving Equation (A.12) due to the construction of $\tilde{V}$. Also, $-T^{*}$ is the most recent time in the past such that $Q_{k,k}(-T^{*} - 1) = 0$.

Now, we derive a lower bound on $\overline{B}^{N}_{(k,k)}(-T^{*}, m)$, which will in turn provide an upper bound of $\Pr(\tilde{V}^{(N)}(k) > m)$. We make use of the priority scheme of SMART in the two dimensional queueing framework, which has been discussed above. An observation was made that area A is higher priority compared to $Q_{k,k}$, and the queues in area B may or may not have higher priority. Thus a simple lower bound on $\hat{B}^{N}_{(k,k)}(-T^{*}, m)$ is the available service assuming that both areas A and B have higher priority. The volume of service that area A requires can be derived using the fact that all queues in area A are empty at time $-T^{*} - 1$, i.e., $Q_{(i,j)}(-T^{*} - 1) = 0$, for all $i \leq k$ and $j \leq k - 1$. The proof follows immediately from Theorem 4.1 in [38], which is stated in the context of priority queues. Additionally, the volume of service that area B requires during interval $(-T^{*}, m)$ is upper bounded by

$(T^* + m + 1)(M - 1)$. This is due to the observation that at most a single partially serviced job can occur in a time slot, and the worst partially serviced job is of size $M - 1$. Thus $\overline{B}^N_{(k,k)}(-T^*, m)$ is lower bounded as follows.

$$\begin{aligned}
\overline{B}^N_{(k,k)}(-T^*, m) &\geq NC(T^* + m + 1) - \sum_{i=1}^{k} \sum_{j=1}^{k-1} Q_{(i,j)}(-T^* - 1) \\
&\quad - \sum_{i=1}^{k-1} iA^N_i(-T^*, m) - (T^* + m + 1)(M - 1) \\
&= NC(T^* + m + 1) - \sum_{i=1}^{k-1} iA^N_i(-T^*, m) \\
&\quad -(T^* + m + 1)(M - 1) \qquad \text{(A.13)}
\end{aligned}$$

From Equation (A.11), Equation (A.12), and Equation (A.13), we have

$$\begin{aligned}
&\Pr\left(\tilde{V}^{(N)}(k) > m\right) \\
\leq\ &\Pr\left[kA^N_k(-T^*, 0) + k - \overline{B}^N_{(k,k)}(-T^*, m) > 0\right] \\
\leq\ &\Pr\left[kA^N_k(-T^*, 0) + \sum_{i=1}^{k-1} iA^N_i(-T^*, m) - NC(T^* + m + 1) + (T^* + m + 1)(M - 1) + k > 0\right] \\
\leq\ &\Pr\left[kA^N_k(-T^*, 0) + \sum_{i=1}^{k-1} iA^N_i(-T^*, m) - NC(T^* + m + 1) + 2(T^* + m + 1)M > 0\right] \\
\leq\ &\Pr\left[kA^N_k(-T^*, 0) + \sum_{i=1}^{k-1} iA^N_i(-T^*, m) - N\left(C - \frac{2M}{N}\right)(T^* + m + 1) > 0\right] \\
\leq\ &\Pr\left[\bigcup_{T\geq 0}\left(kA^N_k(-T, 0) + \sum_{i=1}^{k-1} iA^N_i(-T, m) - N\left(C - \frac{2M}{N}\right)(T + m + 1) > 0\right)\right] \\
\leq\ &\sum_{T\geq 0} \Pr\left[kA^N_k(-T, 0) + \sum_{i=1}^{k-1} iA^N_i(-T, m) - N\left(C - \frac{2M}{N}\right)(T + m + 1) > 0\right].
\end{aligned}$$

Fix any $\epsilon > 0$ and observe that for N large enough, we have $(C - \frac{2M}{N}) > (C - \epsilon)$. Hence,

$$\Pr\left(\tilde{V}^{(N)}(k) > m\right) \leq \sum_{T \geq 0} \Pr\left[kA_k^N(-T,0) + \sum_{i=1}^{k-1} iA_i^N(-T,m)\right.$$
$$\left. -N(C-\epsilon)(T+m+1) > 0\right]. \tag{A.14}$$

Note that Equation (A.14) is the expression for the decay rate of size k jobs in PRI having capacity $C - \epsilon$. Using similar techniques as in [13, 24], it follows that the lower bound of $I_{\tilde{V}_\mu}(k,m)$ is $I_{\overline{V}_{C-\epsilon}}(k,m)$. Applying Lemma A.3.1, we have the lower bound on the decay rate in Equation (4.3). Specifically, we apply the contraction principle, to obtain the closed form expression of $I_{V_{C-\epsilon}}(k,m)$.

Next, we derive the upper bound on the decay rate in Equation (4.3) by deriving a lower bound on $\Pr(V^{(N)}(k) > m)$ and combine it with the result of Lemma A.3.1. We do so by comparing SMART with a priority queueing system which lower bounds the delay experienced by the job.

Consider a PRI system with capacity NC, which we describe again. This system consists of M queues, with jobs of original size k arriving to queue-k. There are strict priority between queues where the queue corresponding to smaller jobs have higher priority. Partially served jobs in this system continue to reside in the same queue and the jobs in each queue are served in a FCFS manner. In comparing the PRI system to SMART, we can think of PRI as SMART where $Q_k = \sum_{i=1}^{k} Q_{k,i}$ and priorities are assigned such that area A has higher priority whereas area B and C have lower priority.

Denote $V^{(N)}(k)$ as the virtual delay of a size k job for PRI. Then event $\{V^{(N)}(k) \leq m\}$ of PRI ensures the event $\{\overline{V}^{(N)}(k) \leq m\}$ of SMART. This comes from the fact that the external arrival to both PRI and SMART are the same but the residual service available to $Q_{k,k}$ in SMART is upper bounded by the residual service for queue-k of PRI. This follows from the fact that the residual service of SMART is the remaining service after servicing of all of area A and possibly a part or all of area B. However, the residual service in PRI is that of after servicing only the area A, and none of area B. Thus PRI provides a lower bound on the virtual delay of a job compared to SMART. In other words, we have $I_{\overline{V}_C}(k,m) \leq I_V(k,m)$ and combining it with Lemma A.3.1, the proof is complete. $\square$

A.4 Proof of Theorem 5.3.1

A.4.1 Lower bound of the decay rate

We start the analysis of LAS by deriving a lower bound on the delay decay rate under LAS. Denote the virtual delay of LAS as $\hat{V}(0)$. We bound the probability that the virtual delay for size k jobs exceeds m, $\Pr(\hat{V}^{(N)}(k) > m)$, as follows.

Lemma A.4.1. *For any $k \in \mathcal{M}$, the virtual delay of size k jobs in LAS satisfies*

$$\begin{aligned} \Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right) \\ &= \Pr\left(kA_k^N(-T^*,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T^*,m) > 0\right), \quad \text{(A.15)} \end{aligned}$$

where $A_k^N(-T,0)$ is the number of size k job arrivals in the interval $(-T,0)$, $\hat{B}_k(-T,m)$ is the service available to Q_k during $(-T,m)$, and T^* is the last time before 0 that $Q_k(-T^*-1) = 0$

The theorem states that a *possible* scenario in which the event $\{\hat{V}^{(N)}(k) > m\}$ could occur is when the total volume of arrivals for size k job before the virtual job and a portion $((k-1)/k)$ of the volume of arrivals for size k job after the virtual job, exceed the available capacity for Q_k .

Proof. Consider Q_k. By the operation of LAS, $\{\hat{V}^{(N)}(k) > m\}$ implies that all jobs in $Q_k(0)$ have not been fully serviced by time m, i.e., $\{kA_k^N(-T^*,0) > \hat{B}_k^N(-T^*,m)\}$. Since adding more jobs makes the event of exceeding the available capacity more probable, we have the following.

$$
\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) - \hat{B}_k^N(-T,0)\right) - \hat{B}_k^N(1,m) > 0\right) \\
&= \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) - \hat{B}_k^N(-T,m)\right) > 0\right) \\
&\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right)
\end{aligned}
$$

Finally, Equation (A.15) follows from Loynes' formula. □

Note that the total available capacity to Q_k in interval $(-T, m)$ with regards to the event $\{\hat{V}^{(N)}(k) > m\}$, i.e. $\hat{B}_k^N(-T, m)$, is the remaining capacity after all the higher priority queues are served in LAS. So, the following holds.

$$
\begin{aligned}
\hat{B}_k^N(-T,m) &\geq NC(T+m+1) - \sum_{i=1}^{k-1} iA_i^N(-T,m) - \sum_{i=1}^{k-1} Q_i(-T-1) \\
&\quad - \sum_{i=k+1}^{M} (k-1)A_i^N(-T,m) - \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(-T-1)
\end{aligned}
\tag{A.16}
$$

Using Lemma A.4.1 and the above, we can derive the following.

Theorem A.4.1. *The virtual decay rate of size k jobs under LAS is lower bounded as follows*

$$
I_{\hat{V}}(k,m) \geq \inf_{T\geq 0}\left[\inf_{\vec{y}:\mathcal{Y}}\left\{\inf_{\vec{x}:\mathcal{X}}\left(\mathfrak{A}_{<k}(\vec{y}) + \mathfrak{A}_k(\vec{y}) + \mathfrak{A}_{>k}(\vec{y})\right)\right\}\right], \tag{A.17}
$$

where $\mathcal{Y}$, $\mathcal{X}$, $\mathfrak{A}_{<k}(\vec{y})$, $\mathfrak{A}_k(\vec{y})$, and $\mathfrak{A}_{>k}(\vec{y})$ are defined as in Theorem 5.3.1.

Proof. Combining Equation (A.16) with Lemma A.4.1 we have the following upper bound on the probability of $\{\hat{V}^{(N)}(k) > m\}$.

$$
\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right) \\
&= \Pr\left(kA_k^N(-T^*,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T^*,m) > 0\right) \\
&\leq \Pr\left(kA_k^N(-T^*,0) + (k-1)A_k^N(1,m) + \sum_{i=1}^{k-1} iA_i^N(-T^*,m) + \right. \\
&\qquad \sum_{i=1}^{k-1} Q_i(-T^*-1) + \sum_{i=k+1}^{M} (k-1)A_i^N(-T^*,m) \\
&\qquad \left. + \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(-T^*-1) - NC(T^*+m+1) > 0\right) \\
&= \Pr\left(kA_k^N(-T^*,0) + (k-1)A_k^N(1,m) + \sum_{i=1}^{k-1} iA_i^N(-T^*,m)\right. \\
&\qquad \left. + \sum_{i=k+1}^{M} (k-1)A_i^N(-T^*,m) - NC(T^*+m+1) > 0\right) \qquad \text{(A.18)} \\
&\leq \Pr\left(\bigcup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) + \sum_{i=1}^{k-1} iA_i^N(-T,m)\right.\right. \\
&\qquad \left.\left. + \sum_{i=k+1}^{M} (k-1)A_i^N(-T,m) - NC(T+m+1)\right) > 0\right) \\
&\leq \sum_{T\geq 0}\Pr\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) + \sum_{i=1}^{k-1} iA_i^N(-T,m)\right. \\
&\qquad \left. + \sum_{i=k+1}^{M} (k-1)A_i^N(-T,m) - NC(T+m+1) > 0\right)
\end{aligned}
$$

where Equation (A.18) follows from the observation that $Q_i(-T^*-1) = 0$ for $1 \leq i \leq k-1$ and $Q_{i,j}(-T^*-1) = 0$ for $k+1 \leq i \leq M$, $0 \leq j \leq k-2$.The justification is similar to that for a priority queueing system. Namely, since the above queues have higher priority than Q_k, when Q_k is empty all higher priority queues must be empty. Applying the contraction principle completes the proof. □

Theorem A.4.1 provides a *possible* scenario in which $\{\hat{V}^{(N)}(k) > m\}$ could occur. To show that this scenario is indeed the most dominant and controls the behavior of the probability, in the next section we will show that the same scenario provides the upper bound on the decay rate of $\{\hat{V}^{(N)}(k) > m\}$.

A.4.2 The delay decay rate (a tight upper bound)

We now develop an upper bound for the decay rate of LAS which is tight with the lower bound derived in Section A.4.1, i.e., we develop the tight lower bound for the tail probability. The main argument for the upper bound is that, using the two dimensional queueing framework, LAS can be viewed as a simple priority queueing system, as was the case with SMART. Thus, using a similar analysis, we can derive the tight upper bound for the decay rate.

Theorem A.4.2. *The probability of the virtual delay for size k jobs in LAS can be lower bounded as*

$$\begin{aligned}\Pr\left(\hat{V}^{(N)}(k) > m\right) \quad \geq \quad & \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1} iA_i^N(-T^*, l) + kA_k^N(-T^*, 0) + (k-1)A_k^N(1, l)\right.\right. \\ & \left.\left. + \sum_{i=k+1}^{M}(k-1)A_i^N(-T^*, l) - NC(T^* + l + 1)\right\} > 0\right) \qquad \text{(A.19)}\end{aligned}$$

where $-T^*$ is the last time before time 0 when $Q_k(-T^* - 1) = 0$.

Proof. As explained in Section A.4.1, there exists a group of queues and arrivals that constitute higher priority compared to the virtual job that arrived at time 0. At time l the volume of jobs from higher priority queues and arrivals is

$$\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(l) + \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(l) + \sum_{j=0}^{k-2} Q_{k,j}(l) + A_k(-T^*, 0). \qquad \text{(A.20)}$$

If the higher priority queues and arrivals with respect to the virtual jobs in Equation (A.20) are never empty at any time during $(0, m)$, then the virtual job is guaranteed not to leave the system in the interval $(0, m)$. Based on this observation we derive a lower bound on $\Pr(\hat{V}^{(N)}(k) > m)$ as follows.

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) \quad &\geq \quad \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(l) + \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(l)\right.\right. \\
&\qquad \left.\left. + \sum_{j=0}^{k-1} Q_{k,j}(l) + A_k(-T^*, 0)\right\} > 0\right) \\
&= \quad \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(-T^* - 1) + \sum_{i=1}^{k-1} iA_i^N(-T^*, l)\right.\right. \\
&\qquad + \sum_{i=k+1}^{M}\sum_{j=0}^{k-1} Q_{i,j}(-T^* - 1) + \sum_{i=k+1}^{M} (k-1)A_i^N(-T^*, l) \\
&\qquad \left.\left. + kA_k^N(-T^*, 0) + (k-1)A_k^N(1, l) - NC(T^* + l + 1)\right\} > 0\right) \\
&= \quad \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1} iA_i^N(-T^*, l) + \sum_{i=k+1}^{M} (k-1)A_i^N(-T^*, l)\right.\right. \\
&\qquad \left.\left. + kA_k^N(-T^*, 0) + (k-1)A_k^N(1, l) - NC(T^* + l + 1)\right\} > 0\right)
\end{aligned}$$

In the calculation above, note that we have argued in Section A.4.1 that at time $-T^* - 1$ all higher priority queues are empty. ◻

Extending the above to obtain a tight lower bound on the decay rate of LAS is difficult without further assumptions on the inputs. We make two assumptions in order to complete the derivation.

Assumption A.4.1. *Let A_{high} denote the sum of all higher priority arrivals described in Theorem A.4.2, then we assume that the corresponding rate function satisfies*

$$I_{A_{high}}^{(-T^*,l)}\left(C(T^* + l + 1) - v\right) < I_{A_{high}}^{(-T^*,0)}\left(C(T^* + 1)\right) \tag{A.21}$$

for $v \in [v^* - \delta, v^* + \delta]$, $v^* > 0$, and $\delta > 0$ sufficiently small.

Assumption A.4.2. *Define*

$$\vec{A}_{high} = \Big(\ldots, A^N_{high}(-T,0), \ldots, A^N_{high}(-1,0), A^N_{high}(0,0)\Big)$$

Then, we assume that the stochastic process $\vec{A}_{high}$ satisfies

$$\Big(\vec{A}_{high} | A^N_{high}(0,0) = 0\Big) \leq_{st} \Big(\vec{A}_{high} | A^N_{high}(0,0) > 0\Big).$$

where $\vec{A}_{high}$ was defined in Assumption A.4.1.

The first assumption is equivalent to the decay rate being additive. Intuitively, a decay rate with the property of additive functionals implies that the occurrence of a rare event in the large deviation framework happens in a straight line. This assumption has been used extensively in large deviation literature [5, 14, 27]. Further, arrival processes that satisfy Assumption A.4.1 include many common processes such as all stationary and Markov dependent processes. Additionally, if the arrival process is of Levy type then the decay rate of the the arrival satisfies Equation (A.21).

The second assumption allows us to show that the virtual delay decay rate is equal to the actual delay decay rate. This assumption essentially says that the arrival process has the property that if there are very few arrivals in a given time slot, there were also very few arrivals in the immediate past (and vice versa). It is a kind of “burstiness” assumption for the source.

We are now ready to complete the proof of Theorem 5.3.1.

Proof. *(of Theorem 5.3.1)* It follows from Theorem A.4.2 and Assumption A.4.1 that $I_{\hat{V}}(k,m)$ is upper bounded by the expression in Equation (5.1) using the same technique as in [14]. Further, applying Theorem A.4.1, we obtain equality. Lastly, using similar arguments as in

[38], we can conclude that the actual delay decay rate is equal to the virtual delay decay rate under Assumption A.4.2, which completes the proof. ◻

A.5 Proof of Theorem 6.3.1

A.5.1 Lower bound on the decay rate

First, note that an upper bound on the probability is a lower bound on the decay rate. We start the analysis of finite-SRPT by deriving an upper bound on the probability of the virtual delay ($V^{(N)}(k)$) exceeding some threshold m, i.e., $\Pr(V^{(N)}(k) > m)$.

Denote $B^{N,k}(a, b)$ as the service capacity available to all the the files in $Q_{k,k}(0)$ including the tagged file during interval (a, b), and $B^N_{i,j}(a, b)$ as the service capacity available to the queue $Q_{i,j}$ during the time-interval (a, b). To illustrate the difference, consider the case where the tagged file of size 3 arrives to the system and is served 2 units at time-slot 2 and the rest is served at time-slot 7. In this case, the service available to all files of size 3 that arrives at time-slot 0 including the tagged file during $(7, 1)$ is $B^{N,3}(7, 1)$. $B^{N,3}(7, 1)$ can be derived by noticing that the tagged file resides in $Q_{3,3}$ during the interval $(2, 1)$, and $Q_{3,1}$ during $(7, 3)$, thus $B^{N,3}(7, 1)$ is the summation of the available service in these queues during their respective time intervals, i.e.,

$$B^{N,3}(7, 1) = B^N_{3,3}(2, 1) + B^N_{3,1}(7, 3).$$

Let us focus our attention on $Q_{k,k}$. By the operation of finite-SRPT, $\{V^{(N)}(k) > m\}$ implies that all files in $Q_{k,k}(0)$ have not been fully serviced by time m. In other words, $\Pr(V^{(N)}(k) > m) \leq \Pr(Q_{k,k}(0) > B^{N,k}(m, 1))$. Denote the lower bound of $B^{N,k}(m, 1)$ as $B^{N,k}_{LB}(m, 1)$. Then the following holds.

$$\begin{aligned} \Pr(V^{(N)}(k) > m) &\leq \Pr(Q_{k,k}(0) > B^{N,k}(m,1)) \\ &\leq \Pr(Q_{k,k}(0) > B^{N,k}_{LB}(m,1)) \end{aligned} \tag{A.22}$$

To connect with the simple case, we show that Equation (6.5) can revised in the same manner as Equation (A.22) as follows.

$$\Pr(V^{(N)}(k) \geq m) \leq$$

$$\begin{cases} \Pr\Big(Q_{k,k}(0) \geq \Big(NC(m+1) - \sum_{i=1}^{k-1} A_i(m,1) \\ \quad - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0)\Big)_{LB}\Big), & k \leq D \\ \Pr\Big(Q_{k,k}(0) \geq \Big(NC(m+1) - \sum_{i=1}^{k-1} A_i(T_1,1) \\ \quad - \sum_{i=1}^{k-D-1} A_i(m, T_1+1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0)\Big)_{LB}\Big), & D < k \leq 2D \end{cases} \tag{A.23}$$

where $B^{N,k}_{LB}(m,1)$ is equal to $(NC(m+1) - \sum_{i=1}^{k-1} A_i(m,1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0))_{LB}$ for $k \leq D$ and $(NC(m+1) - \sum_{i=1}^{k-1} A_i(T_1,1) - \sum_{i=1}^{k-D-1} A_i(m, T_1+1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0))_{LB}$ for $D < k \leq 2D$. The arguments in Section 6.2 provides a clear cut way to find $B^{N,k}_{LB}(m,1)$ by providing upper bounds on the volume of files in higher priority queues and higher priority file arrivals for the simple case of $D < M \leq 2D$.

However, we require the solution for the general case. To extend the result to a more general case, we make use of the idea that $B^{N,k}(m,1)$ can be broken down into multiple $B^N_{i,j}(a,b)$'s. In particular, we have $B^{N,k}(m,1) = B^N_{k,k}(m,1)$ for $k \leq D$ and $B^{N,k}(m,1) = B^N_{k,k}(T_1,1) + B^N_{k,k-D}(T_2, T_1+1)$, where $B^N_{k,k}(m,1) \geq NC(m+1) - \sum_{i=1}^{k-1} A_i(m,1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0)$ for $k \leq D$ and

$$
\begin{aligned}
B^N_{k,k}(T_1, 1) + B^N_{k,k-D}(m, T_1 + 1) \geq NC(m+1) - \sum_{i=1}^{k-1} A_i(T_1, 1) \\
- \sum_{i=1}^{k-D-1} A_i(m, T_1 + 1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0),
\end{aligned}
$$

for $D \leq k < 2D$.

This idea can be extended to the general case in the following manner. Denote T_i as the time-slot in which the tagged file receives its i'th D units of service. Then it is clear that a job of size k requires $\lceil k/D \rceil$ rounds of service, and will leave the system at time-slot $T_{\lceil k/D \rceil}$. We denote $r_1 + 1$ as the last round of service in which the tagged file leaves the system, i.e., $r_1 = \lceil k/D \rceil - 1$, thus $T_{r_1+1} = m$.

Now we characterize the lower bound of $B^{N,k}(m, 1)$. Note that the total available capacity to all higher priority files in interval $(m, 1)$ with regards to the event $\{V^{(N)}(k) > m\}$, i.e., $B^{N,k}(m, 1)$, is the remaining capacity after all the higher priority queues are served in finite-SRPT. Using similar argument as the simple $D < M \leq 2D$ case, we have

$$
\begin{aligned}
B^{N,k}(m, 1) &= B_{k,k-r_1D}(T_{r_1+1}, T_{r_1} + 1) + \ldots + B_{k,k-D}(T_2, T_1 + 1) + B_{k,k}(T_1, 1) \\
&\geq NCT_{r_1+1} - \sum_{i=1}^{k-r_1D-1} iA^N_i(T_{r_1+1}, T_{r_1} + 1) - \ldots - \sum_{i=1}^{k-D-1} iA^N_i(T_2, T_1 + 1) \\
&\quad - \sum_{i=1}^{k-1} iA^N_i(T_1, 1) - T_{r_1+1}(M-1) - \sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0). \qquad \text{(A.24)}
\end{aligned}
$$

We complete the lower bound by deriving an upper bound on $\sum_{i=1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(0)$. Section 6.2 reports a suitable upper bound for the simple case of $D < M \leq 2D$ in Equation (6.6) for $k \leq D$ and in Equation (6.7) for $D < k \leq 2D$. Lastly, the upper bound of $\Pr(V^{(N)}(k) \geq m)$ can completely characterized by noting that by definition

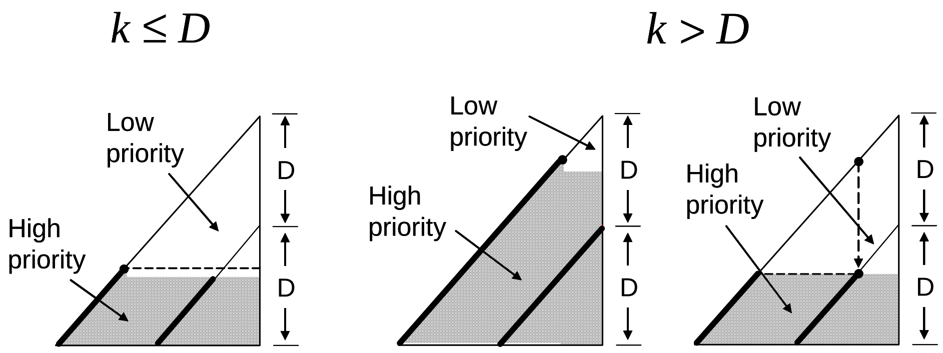


Figure A.3: Priority scheme for a tagged job of size k, where $D < M \le 2D$. The higher priority area is composed of typical higher priority jobs represented as the thick diagonal lines and atypical jobs are in the rest of the shaded area that represent the atypical jobs.

$$Q_{k,k}(0) = kA_k^N(0, T_0 + 1). \tag{A.25}$$

To extend this result to the general case we make the following arguments. Higher priority files with respect to $Q_{k,k}(0)$ can be separated into two different groups: the typical and the atypical higher priority files. In other words,

$$\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) = \sum_{i=1}^{k}\sum_{j\in\mathcal{A}-\{k\}} Q_{i,j}(0) + \sum_{i=1}^{k}\sum_{j\in\{1,2,...,k\}-\mathcal{A}} Q_{k,j}(0), \tag{A.26}$$

where $\mathcal{A} = \{i, i-D, i-2D, ..., i-sD\}$ and $s = \lceil \frac{i}{D} \rceil - 1$. Note that $\sum_{i=1}^{k}\sum_{j\in\mathcal{A}-\{k\}} Q_{i,j}(0)$ is the volume of typical higher priority files in the thick diagonal strip and $\sum_{i=1}^{k}\sum_{j\in\{1,2,...,k\}-\mathcal{A}} Q_{k,j}(0)$ is the total volume of atypical files in the higher priority queues as shown in Figure A.3.

Based on similar arguments as in Section 6.2, the volume of all files in the higher priority queues in Equation (A.26) can be upper bounded by

$$\begin{aligned}\sum_{i=1}^{k}\sum_{j\in\mathcal{A}-\{k\}} Q_{i,j}(0) \quad \leq \quad & \sum_{i=1}^{k-1} iA_i^N(0,T_0+1)+\sum_{i=1}^{k-D} iA_i^N(T_0,T_{-1}+1)+\dots \\ & + \sum_{i=1}^{k+r_2D} iA_i^N(T_{r_2+1},T_{r_2}+1)-T_{r_2}(M-1)-NCT_{r_2},\end{aligned} \tag{A.27}$$

and

$$\sum_{i=1}^{k}\sum_{j\in\{1,2,\ldots,k\}-\mathcal{A}} Q_{k,j}(0) \leq -T_{r_2}(M-1), \tag{A.28}$$

where $r_2 = \lceil \frac{k}{D} \rceil - \lceil \frac{M}{D} \rceil$, and $0 > T_{-1} > T_{-2} > \ldots > T_{r_2}$. The upper bound described in Equation (A.27) is the result of the fact that $\sum_{i=1}^{k} iA_i^N(0,T_0+1)$ contributes to the upper most strip and $\sum_{i=1}^{k-D} iA_i^N(T_0,T_{-1}+1)$ results in files in the next lower strip and so on.

Next, the upper bound in Equation (A.28) corresponds to the maximum volume of the atypical higher priority files. The maximum volume of such files in interval $(0, T_{r_2})$ is $(1-T_{r_2})(M-1)$ since at most only one file can become atypical in a time-slot and the largest possible atypical file size is $M-1$.

Combining Equation A.27 and Equation A.28, we have

$$\begin{aligned}\sum_{i=1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(0) \quad \leq \quad & \sum_{i=1}^{k-1} iA_i^N(0,T_0+1)+\sum_{i=1}^{k-D} iA_i^N(T_0,T_{-1}+1)+\dots \\ & + \sum_{i=1}^{k+r_2D} iA_i^N(T_{r_2+1},T_{r_2}+1)-2T_{r_2}(M-1)-NCT_{r_2},\end{aligned} \tag{A.29}$$

Combining the result of Equation (A.29) and Equation (A.24), we have

$$\begin{aligned}
B_k^N(m,1) \quad \geq \quad & NC(m+1-T_{r_2}) - \sum_{i=1}^{k-r_1D-1} iA_i^N(T_{r_1+1}, T_{r_1}+1) - \ldots \\
& - \sum_{i=1}^{k-D-1} iA_i^N(T_2, T_1+1) - \sum_{i=1}^{k-1} iA_i^N(T_1, 1) - \sum_{i=1}^{k-1} iA_i^N(0, T_0+1) \\
& - \sum_{i=1}^{k-D} iA_i^N(T_0, T_{-1}+1) - \ldots - \sum_{i=1}^{k+r_2D} iA_i^N(T_{r_2+1}, T_{r_2}+1) \\
& -2(m+1-T_{r_2})(M-1). \qquad\qquad \text{(A.30)}
\end{aligned}$$

Lastly, applying the results from Equations (A.30), and Equation (A.25), to Equation (A.22), we have the following upper bound on the probability of $\{V^{(N)}(k) > m\}$.

$$\begin{aligned}
\Pr(V^{(N)}(k) > m) \quad \leq \quad & \Pr(Q_{k,k}(0) > B^{N,k}(m,1)) \\
\leq \quad & \Pr\Bigg(\sum_{i=1}^{k-r_1D-1} iA_i^N(m, T_{r_1}+1)) + \ldots \sum_{i=1}^{k-D-1} iA_i^N(T_2, T_1+1) + \\
& + \sum_{i=1}^{k-1} iA_i^N(T_1, 1) + \sum_{i=1}^{k} iA_i^N(0, T_0+1) \\
& + \sum_{i=1}^{k+D} iA_i^N(T_0, T_{-1}+1) + \ldots + \sum_{i=1}^{k-r_2D} iA_i^N(T_{r_2+1}, T_{r_2}+1) \\
& +2(m+1-T_{r_2})(M-1) - NC(m+1-T_{r_2}) > 0) \\
= \quad & \Pr\left(\sum_{j=r2}^{r1} \mathcal{A}_j(T_{j+1}, T_j+1) - N\left(C - \frac{2(M-1)}{N}\right)(m+1+T_{r_2}) > 0 \right) \\
\leq \quad & \Pr\left(\bigcup_{\vec{T}:\mathcal{T}} \left(\sum_{j=r2}^{r1} \mathcal{A}_j(T_{j+1}, T_j+1) - N\left(C - \frac{2(M-1)}{N}\right)(m+1+T_{r_2}) > 0 \right) \right) \\
\leq \quad & \sum_{\vec{T}:\mathcal{T}} \Pr\left(\sum_{j=r2}^{r1} \mathcal{A}_j(T_{j+1}, T_j+1) - N\left(C - \frac{2(M-1)}{N}\right)(m+1+T_{r_2}) > 0 \right)
\end{aligned}$$

where condition $\mathcal{T}$ states that $m = T_{r_1} > \ldots > T_1 > 0 > T_0 > \ldots > T_{r_2}$, and $\mathcal{A}_j(T_{j+1}, T_j+1)$ is defined as

$$\mathcal{A}_j(T_{j+1}, T_j + 1) =$$

$$\begin{cases} \sum_{i=1}^{k-jD-1} iA_i(T_{j+1}, T_j + 1) & j \in \{r_1 + 1, r_1, \ldots, 2, 1\} \\ \sum_{i=1}^{k-1} iA_i(T_1, 1) + \sum_{i=1}^{k} iA_i(0, T_0 + 1) & j = 0 \\ \sum_{i=1}^{k-jD} iA_i(T_{j+1}, T_j + 1) & j \in \{-1, -2, \ldots, r_2 + 1, r_2\}. \end{cases}$$

Fix any $\epsilon > 0$ and observe that for N large enough, we have $\left(C - \frac{2(M-1)}{N}\right) > (C - \epsilon)$. Applying the contraction principle we have that the delay decay rate of finite-SRPT, $I_V(k, m)$, is lower bounded by $I^{C-\epsilon}(k, m)$ for any $\epsilon > 0$. Where $I^S(k, m)$ is defined as

$$I^S(k, m) = \inf_{\vec{T}:\mathcal{T}} \left[\inf_{\vec{y}:\mathcal{Y}} \left\{ \sum_{j=1}^{M} I_{A_j^N(T_{i+1}, T_i+1)} \left(y_j^{(T_{i+1}, T_i+1)} \right) \right\} \right], \tag{A.31}$$

where condition $\mathcal{T}$ states that $m = T_{r_1+1} > T_{r_1} > \ldots > T_1 > 0 > T_0 \ldots > T_{r_2}$, and $\mathcal{Y}$ states that

$$\sum_{i=1}^{r_1} \sum_{j=1}^{k-iD-1} j y_j^{(T_{i+1}, T_i+1)} + \sum_{j=1}^{k-1} j y_j^{(T_1, 1)} + \sum_{j=1}^{k} j y_j^{(0, T_0+1)} \\ + \sum_{i=r_2}^{-1} \sum_{j=1}^{k-iD} j y_j^{(T_{i+1}, T_i+1)} = S(m + 1 + T_{r_2}),$$

where $y_j^{(T_{i+1}, T_i+1)}, y_j^{(T_1, 1)}, y_j^{(0, T_0+1)} \geq 0$ for all i, $j \geq 0$.

A.5.2 Upper bound of the decay rate

We now derive an upper bound for the delay decay rate of finite-SRPT which is arbitrarily tight with the lower bound derived in Section A.5.1. This is done by finding a suitable lower bound on the probability of delay, i.e., $\Pr(V^{(N)}(k) > m)$. The main argument is based on the idea that by using the two dimensional queueing framework, finite-SRPT can be viewed as a simple priority queueing system. Then by using similar analysis as the priority queueing system an upper bound can be derived.

As explained in Section 6.2, there exists a group of files that constitute higher priority compared to the tagged size k file that arrives at time 0. Unlike Section A.5.1, files that are absolutely required to be served before the tagged file corresponds to the lower bound on the volume of higher priority files. A lower bound for the simple $D < M \leq D$ case is derived in Section 6.2, and similarly a lower bound for the general case can be derived as

$$\sum_{j=r2}^{r1} \mathcal{A}_j(T_{j+1}, T_j + 1) - (m + 1 - T_{r_2})(M - 1). \tag{A.32}$$

However, consider the following scenario. Assume that time l satisfies $T_{r_1+1} \leq l \leq T_{r_1}$. This implies that at time l the tagged file has received only $k - 1$ units but have not received the last D units of service for it to completely leave the system. Then at time l all files of higher priority are

$$\sum_{j=r2}^{r1} \mathcal{A}_j(T_{j+1}, T_j + 1) - (m + 1 - T_{r_2})(M - 1), \tag{A.33}$$

assuming $T_{r_1+1} = l$. If all the higher priority files with respect to the tagged size k files have not left the system in any time during the interval $(0, m)$, then it is guaranteed that the tagged file did not leave the system in the interval $(0, m)$. Based on this observation, we derive a lower bound on $\Pr(V^{(N)}(k) > m)$ as follows:

$$\Pr(V^{(N)}(k) > m)$$

$$\geq \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{j=r2}^{r1}\mathcal{A}_j(T_{j+1},T_j+1)-N\left(C+\frac{(M-1)}{N}\right)(l+1-T_{r_2})\right\}\right), \quad \text{(A.34)}$$

assuming $T_{r_1+1}=l$. Extending the above result to obtain a tight lower bound on the decay rate of finite-SRPT is difficult without further assumptions on the input. We make the following assumption on the arrival process in order to complete the derivation.

Assumption A.5.1. *Let A_{high} denote the sum of all higher priority arrivals in any session of (T_{i+1},T_i+1) with respect to the tagged size k file as described in Equation A.33, then we assume that the corresponding rate function satisfies*

$$I_{A_{high}}^{(T_{i+1}+l,T_i+1)}(C(T_{i+1}-T_i+l)-v) < I_{A_{high}}^{(T_{i+1},T_i+1)}(C(T_{i+1}-T_i))$$

for $v\in[v^*-\delta,v^*+\delta]$, $v^*>0$, and $\delta>0$ sufficiently small.

The following explanation of the assumptions is an excerpt from Chapter A.4 which is included for completeness. The first assumption is equivalent to the decay rate being additive. Intuitively, a decay rate with the property of additive functionals implies that the occurrence of a rare event in the large deviation framework happens in a straight line. This assumption has been used extensively in large deviation literature [5, 14, 27]. Further, arrival processes that satisfy Assumption A.5.1 include many common processes such as all stationary and Markov dependent processes. Additionally, if the arrival process is of Levy type then the decay rate of the the arrival satisfies Assumption A.5.1.

Using the same technique as in [14] on Equation (A.34), we have that $I^{C-\epsilon}(k,m)$ as an arbitrarily tight upper bound on the delay decay rate of finite-SRPT based on Assumption A.5.1. Combining with the previous lower bound, we obtain the result for virtual delay. This completes the proof.

A.6 Proof of Theorem 7.4.1

The basic idea of the proof of Theorem 7.4.1 is the following (ignoring the slight adjustment required for size k jobs). The explanation described here is only for better understanding and the actual proof considers the adjustment. Denote $[2DQ/i]^d$ and $[A^N(T_{i+1}, T_i + 1)]^d$ as d units of all jobs that arrive to $2DQ/i$ and sum of d units of all jobs in $A^N(T_{i+1}, T_i + 1)$ respectively. Then, we have

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\approx \Pr\Big([2DQ/k]^1 + \ldots + [2DQ/0]^{k-1} + Q^0(T_0) > NC(m+1)\Big) \\
&\approx \Pr\Big([2DQ/k]^1 + \ldots + [2DQ/0]^{k-1} + [2DQ/-1]^k \\
&\qquad + \ldots + [2DQ/k-M]^M > NC(T_* + m + 1)\Big) \\
&= \Pr\Big([A^N(T_{k+1}, T_k + 1)]^1 + \ldots + [A^N(T_1, T_0 + 1)]^{k-1} + [A^N(T_0, T_{-1} + 1)]^k \\
&\qquad + \ldots [A^N(T_{k-M+1}, T_\star + 1)]^k > NC(T_* + m + 1)\Big).
\end{aligned}$$

Based on this intuitive argument the precise proof of the derivation is provided in the following sections.

A.6.1 Lower bound of the delay decay rate of discrete PS

We start the analysis of discrete PS by deriving an upper bound on the probability of virtual delay of size k job exceeding some threshold m, i.e., $\Pr(\hat{V}^{(N)}(k) > m)$. Denote $\hat{B}_k^N(a, b)$ as the service capacity available to all jobs with original size k during the interval (a, b).

Lemma A.6.1. *For any $k \in \mathcal{M}$, the virtual delay of size k jobs in discrete PS satisfies*

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{\vec{T}:\mathcal{T}}\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}, T_i + 1) - \hat{B}_k^N(m, T_{k-M} + 1)\right) > 0\right) \\
&= \Pr\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}^*, T_i^* + 1) - \hat{B}_k^N(m, T_{k-M} + 1) > 0\right)
\end{aligned}$$

where condition $\mathfrak{T}$ *states that* $m \geq T_k \geq T_{k-1} \geq \ldots T_1 \geq 0 \geq T_0 \ldots \geq T_{k-M+1} \geq T_{k-M} = T_*$, T_j^* *are the time-slots for each cycles that supremize the above equation. Further,* $\mathcal{A}_k^N(T_{i+1}, T_i + 1)$ *is defined as*

$$\mathcal{A}_k^N(T_{i+1}, T_i + 1) = \begin{cases} kA_k^N(T_{i+1}, T_i + 1) & \textit{for } k - M \leq i \leq -1 \\ (k-1)A_k^N(T_1, 1) + kA_k^N(0, T_0 + 1) & \textit{for } i = 0 \\ (k-i)A_k^N(T_{i+1}, T_i + 1) & \textit{for } 1 \leq i \leq k - 1. \end{cases}$$

The theorem states that a *possible* scenario in which the event $\{\hat{V}^{(N)}(k) > m\}$ could occur is when the total *relevant* volume of arrivals for size k job across all future and past 2DQs for size k jobs exceed the available capacity to all jobs of size k.

Proof. Consider Q_k^0. By the operation of discrete PS, $\{\hat{V}^{(N)}(k) > m\}$ implies that all jobs in $Q_k^0(0)$ have not been fully serviced by time m, i.e., $\{Q_k^0(0) > \hat{B}_k^N(m, 1)\}$. Since adding more jobs makes the event of exceeding the available capacity more probable, we have the following.

$$\begin{aligned} \Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(Q_k^0(0) > \hat{B}_k^N(m, 1)\right) \\ &= \Pr\left(\sup_{T<0}\left(kA_k^N(0, T) - \hat{B}_k^N(0, T)\right) - \hat{B}_k^N(m, 1) > 0\right) \\ &= \Pr\left(kA_k^N(0, T_* + 1) - \hat{B}_k^N(m, T_* + 1) > 0\right) \\ &= \Pr\left(kA_k^N(0, T_{k-M} + 1) - \hat{B}_k^N(m, T_{k-M} + 1) > 0\right) \\ &\leq \Pr\left(kA_k^N(T_1, T_{k-M} + 1) - \hat{B}_k^N(m, T_{k-M} + 1) > 0\right) \\ &\leq \Pr\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}, T_i + 1) - \hat{B}_k^N(m, T_{k-M} + 1) > 0\right), \end{aligned}$$

for any $\overrightarrow{T} : \mathfrak{T}$. Thus Lemma A.6.1 holds. □

Note that the total available capacity to all jobs of size k in interval (m, T) with regards to the event $\{\hat{V}^{(N)}(k) > m\}$, i.e. $\hat{B}_k^N(m, T)$, is the remaining capacity after all the higher priority queues are served in discrete PS. So, the following inequality holds for $T < 0$.

$$\hat{B}_k^N(m, T) \geq NC(T + m + 1) - \sum_{i=k-M}^{k-1} \mathfrak{A}_{\neq k}^N(T_{i+1}, T_i + 1) - \sum_{j=1, j\neq k}^{M} Q_i(T_{k-M}), \tag{A.35}$$

where $\mathfrak{A}^N(T_{i+1}, T_i + 1)$ is defined as

$$\mathfrak{A}^N(T_{i+1}, T_i + 1) =$$

$$\begin{cases} \sum_{j=1}^{k-i} jA_j^N(T_{i+1}, T_i + 1) & \\ + \sum_{j=k-i+1}^{M} (k - i)A_j^N(T_{i+1}, T_i + 1) & \text{for } i \neq 0 \\ & \\ \sum_{j=1}^{k-1} jA_j^N(T_1, 1) + \sum_{j=k}^{M} (k - 1)A_j^N(T_1, 1) & \\ + \sum_{j=1}^{k} jA_j^N(0, T_0 + 1) + \sum_{j=k+1}^{M} kA_j^N(0, T_0 + 1) & \text{for } i = 0 \end{cases}$$

and correspondingly $\mathfrak{A}_{\neq k}^N(T_{i+1}, T_i + 1)$ is $\mathfrak{A}^N(T_{i+1}, T_i + 1)$ without the term corresponding to $j = k$. Using Lemma A.6.1 and the lower bound on $\hat{B}_k^N(m, T)$ in Equation (A.35), we derive the following theorem which specifies a lower bound of the delay decay rate of discrete PS, i.e., upper bound on the probability.

Theorem A.6.1. *The decay rate of size k jobs under discrete PS is lower bounded as follows*

$$I_{\hat{V}}(k, m) \geq \inf_{\vec{T}:\mathcal{T}} \left[\inf_{\vec{y}:\mathcal{Y}} \left\{ \sum_{j=1}^{M} I_{A_j^N(T_{i+1}, T_i+1)} \left(y_j^{(T_{i+1}, T_i+1)} \right) \right\} \right], \tag{A.36}$$

where condition $\mathfrak{T}$ *states that* $m \geq T_k \geq T_{k-1} \geq \ldots T_1 \geq 0 \geq T_0 \ldots \geq T_{k-M+1} \geq T_{k-M} = T_*$, *and* $\mathfrak{Y}$ *states that*

$$\begin{aligned}&\sum_{i=k-M, i\neq 0}^{k-1}\left(\sum_{j=1}^{k-i} j y_j^{(T_{i+1}, T_i+1)} + \sum_{j=k-i+1}^{M}(k-i) y_j^{(T_{i+1}, T_i+1)}\right) + \sum_{j=1}^{k} j y_j^{(0, T_0+1)} \\ &+ \sum_{j=k+1}^{M} k y_j^{(0, T_0+1)} + \sum_{j=1}^{k-1} j y_j^{(T_1, 1)} + \sum_{j=k+1}^{M}(k-1) y_j^{(T_1, 1)} = C(T_{k-M} + m + 1),\end{aligned}$$

where $y_j^{(T_{i+1}, T_i+1)}, y_j^{(T_1, 1)}, y_j^{(0, T_0+1)} \geq 0$ *for all* $i, j \geq 0$.

Proof. Combining Equation (A.35) with Lemma A.6.1 we have the following upper bound on the probability of $\{\hat{V}^{(N)}(k) > m\}$.

$$\begin{aligned}\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{\vec{T}:\mathfrak{T}}\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}, T_i + 1) - \hat{B}_k^N(m, T_{k-M} + 1)\right) > 0\right) \\ &= \Pr\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}^*, T_i^* + 1) - \hat{B}_k^N(m, T_{k-M} + 1) > 0\right) \\ &\leq \Pr\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}^*, T_i^* + 1) + \sum_{i=k-M}^{k-1} \mathfrak{A}_{\neq k}^N(T_{i+1}^*, T_i^* + 1)\right. \\ &\qquad \left. + \sum_{j=1, j\neq k}^{M} Q_i(T_{k-M}) - NC(T_{k-M} + m + 1) > 0\right) \\ &\leq \Pr\left(\sum_{i=k-M}^{k-1} \mathcal{A}_k^N(T_{i+1}^*, T_i^* + 1) + \sum_{i=k-M}^{k-1} \mathfrak{A}_{\neq k}^N(T_{i+1}^*, T_i^* + 1)\right. \\ &\qquad \left. - NC(T_{k-M} + m + 1) > 0\right) \qquad \text{(A.37)} \\ &= \Pr\left(\sum_{i=k-M}^{k-1} \mathfrak{A}^N(T_{i+1}^*, T_i^* + 1) - NC(T_{k-M} + m + 1) > 0\right) \\ &\leq \Pr\left(\bigcup_{\vec{T}:\mathfrak{T}}\left(\sum_{i=k-M}^{k-1} \mathfrak{A}^N(T_{i+1}, T_i + 1) - NC(T_{k-M} + m + 1) > 0\right)\right) \\ &\leq \sum_{\vec{T}:\mathfrak{T}} \Pr\left(\sum_{i=k-M}^{k-1} \mathfrak{A}^N(T_{i+1}, T_i + 1) - NC(T_{k-M} + m + 1) > 0\right)\end{aligned}$$

where Equation (A.37) follows from the observation that $Q_i(T_*) = 0$ for all $1 \le i \le M$ by definition. Applying the contraction principle completes the proof. $\square$

Theorem A.6.1 provides a *possible* scenario in which $\{\hat{V}^{(N)}(k) > m\}$ could occur. To show that this scenario is indeed the most dominant and controls the most dominant decaying behavior of the probability, we show that the same equation provides an upper bound on the decay rate of $\{\hat{V}^{(N)}(k) > m\}$.

A.6.2 Tight upper bound of the delay decay rate of discrete PS

We now develop an upper bound for the decay rate of discrete PS which is tight with the lower bound derived in Section A.6.1. The main argument for the upper bound is that by using the 2DQ framework with cycles, discrete PS can be viewed as a simple priority queueing system. Thus, using a similar analysis of the well known priority queueing system, we can derive the tight upper bound for the decay rate.

Theorem A.6.2. *The probability of the virtual delay for size k jobs in LAS can be lower bounded as*

$$\Pr\left(\hat{V}^{(N)}(k) > m\right) \ge \Pr\left(\inf_{0 \le T_k \le m} \left\{\sum_{i=k-M}^{k-1} \mathfrak{A}^N(T^*_{i+1}, T^*_i + 1) - NC(T^*_{k-M} + l + 1)\right\} > 0\right),$$

where T^*_i are the optimizing time-slots for each cycle, and they satisfy $m \ge T_k \ge T_{k-1} \ge \ldots T_1 \ge 0 \ge T_0 \ldots \ge T_{k-M+1} \ge T_{k-M} = T_*$.

Proof. As explained in Section 7.4, there exists a group of job arrivals that constitute higher priority compared to the tagged job. If the higher priority queues and arrivals with respect to the virtual job are never empty at any time during $(0, m)$, then the virtual job is guaranteed not to leave the system in the interval $(0, m)$. Based on this observation we derive a lower bound on $\Pr(\hat{V}^{(N)}(k) > m)$ as follows.

$$\begin{aligned}\Pr\left(\hat{V}^{(N)}(k) > m\right) &\geq \Pr\left(\inf_{0\leq T_k\leq m}\left\{\sum_{i=k-M}^{k-1}\mathfrak{A}^N(T^*_{i+1}, T^*_i + 1)\right.\right.\\&\quad\left.\left. + \sum_{i=1}^{j}\sum_{j=1}^{M} Q^{k-M-1}_{i,j}(T_\star) - NC(T^*_{k-M} + l + 1)\right\} > 0\right)\\&= \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=k-M}^{k-1}\mathfrak{A}^N(T^*_{i+1}, T^*_i + 1) - NC(T^*_{k-M} + l + 1))\right\} > 0\right),\end{aligned}$$

where the last equality follows from the fact that at time-slot $T_\star$ the system is empty. □

To obtain a tight lower bound on the decay rate of discrete PS is difficult without further assumptions on the input. Thus, we make the assumption Assumption 7.4.1 described in Section 7.4. We are now ready to complete the proof of Theorem 7.4.1.

Proof. (of Theorem 7.4.1) It follows from Theorem A.6.2 and Assumption 7.4.1 that $I_{\hat{V}}(k, m)$ is upper bounded by the right hand expression in Equation (7.1) using the same technique as in [14]. Further, applying Theorem A.6.1, we obtain equality. □

Bibliography

[1] S. Aalto, U. Ayesta, and E. Nyberg-Oksanen. Two-level processor-sharing scheduling disciplines: mean delay analysis. In *Proc. of ACM Sigmetrics/Performance*, pages 97–105, 2004.

[2] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *Proceeding of the ACM Sigmetrics*, Philadelphia, PA, April 1996.

[3] M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[4] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. In *Proc. of ACM Sigmetrics/Performance*, pages 279–290, 2001.

[5] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis. Asymptotic buffer overflow probabilities in multiclass multiplexers: An optimal control approach. *IEEE Trans. on Auto. Control*, 43:315–335, 1998.

[6] S. C. Borst, O. J. Boxma, and R. Núñez-Queija. The equivalence between processor sharing and service in random order. *Operations Research Letters*, 31:254–262, 2003.

[7] S. C. Borst, O. J. Boxma, R. Núñez-Queija, and A. P. Zwart. The impact of the service discipline on delay asymptotics. *Performance Evaluation*, 54(2):175–206, October 2003.

[8] D. Botvich and N. Duffield. Large deviations, economies of scale, and the shape of the loss curve in large multiplexers. *Queueing Systems*, 20:293–320, 1995.

[9] E. G. Coffman, R. R. Muntz, and H. Trotter. Waiting time distribution for processor sharing systems. *Journal of the Association of Computing Machinery*, 17:123–130, 1970.

[10] C. Coubercoubetis and R. Weber. Buffer overflow asymptotics for a buffer handling many traffic sources. *J. Appl. Prob.*, 33(3):886–903, 1996.

[11] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[12] M. E. Crovella, M. S. Taqqu, and A. Bestavros. *Heavy-tailed probability distributions in the World Wide Web: A Practical Guide To Heavy Tails*. Chapman and Hall, New York, 1998.

[13] S. Delas, R. Mazumdar, and C. Rosenberg. Cell loss asymptotics for buffers handling a large number of independent stationary sources. In *Proc of IEEE Infocom*, volume 2, pages 551–558, 1999.

[14] S. Delas, R. Mazumdar, and C. Rosenberg. Tail asymptotics for HOL priority queues handling a large number of independent stationary sources. *Queue. Sys. Thry. and App.*, 40(2):183–204, 2002.

[15] J. L. V. den Berg. Sojourn times in feedback and processor-sharing queues. *PhD thesis, Rijksuniversiteit Utrecht*, 1990.

[16] F. Guillemin, P. Robert, and B. Zwart. Tail asymptotics for processor sharing queues. *Advances in Applied Probability*, 36:525–543, 2004.

[17] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Implementation of SRPT scheduling in web servers. *ACM Trans. on Comp. Sys.*, 21(2), May 2003.

[18] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, May 2003.

[19] M. Harchol-Balter, K. Sigman, and A. Wierman. Asymptotic convergence of scheduling policies with respect to slowdown. *Performance 2002. IFIP WG 7.3 International Symposium on Computer Modeling, Measurement and Evaluation*, 49(1):241–256, September 2002.

[20] P. Jelenković and P. Momcilović. Large deviation analysis of subexponential waiting time in a processor sharing queue. *Math. Oper. Res.*, 28:587–608, 2003.

[21] F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley, New York, NY, 1976.

[22] L. Kleinrock. Analysis of a time–shared processor. *Naval Research Logistics Quarterly*, 11:59–73, 1964.

[23] L. Kleinrock. *Queueing Systems, Volume 1: Theory*. Wiley–Interscience, New York, 1975.

[24] N. Likhanov and R. Mazumdar. Cell loss asymptotics for buffers fed with a large number of independent stationary sources. *Journal of Applied Probability*, 36:86–96, 1999.

[25] M. Mandjes and M. Nuyens. Sojourn time in the M/G/1 FB queue with light-tailed service times. *Prob. in the Eng. and Info. Sci.*, 19:351–361, 2005.

[26] M. Mandjes and B. Zwart. Large deviations for sojourn times in processor sharing queues. *Queueing Systems*, 52:237–250, 2006.

[27] L. Massoulie. Large deviations estimates for polling and weighted fair queueing service systems. *Adv. Perf. Anal.*, 2(2):103–128, 1999.

[28] D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Improving preemptive prioritization via statistical characterization of OLTP locking. In *Int. Conf on Data Engineering*, 2005.

[29] R. Nunez-Queija. Processor-sharing models for integrated-services networks. *PhD thesis, Eindhoven University*, 2000.

[30] M. Nuyens, A. Wierman, and B. Zwart. Preventing large sojourn times using SMART scheduling. *Under Submission*, 2005.

[31] M. Nuyens and B. Zwart. A large-deviation analysis of GI/GI/1 SRPT queue. *Under Submission*, 2005.

[32] I. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *Proc. of ACM Sigmetrics*, 2003.

[33] I. Rai, G. Urvoy-Keller, M. Vernon, and E. Biersack. Performance modeling of LAS based scheduling in packet switched networks. In *Proc. of ACM Sigmetrics/Performance*, 2004.

[34] M. Rawat and A. Kshemkalyani. SWIFT: scheduling in web servers for fast response time. In *Symp. on Net. Comp. and App.*, 2003.

[35] R. Righter and J. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Prob. in the Eng. and Info. Sci.*, 3:967–978, 1989.

[36] R. Righter, J. Shanthikumar, and G. Yamazaki. On external service disciplines in single stage queueing systems. *Journal of Applied Probability*, 27:409–416, 1990.

[37] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.

[38] S. Shakkottai and R. Srikant. Many-sources delay asymptotics with applications to priority queues. *Queueing Systems: Theory and Applications*, 39:183–200, October 2001.

[39] W. Stalling. *Operating Systems*. Second Edition, Prentice Hall, 1995.

[40] M. Vojnovic, J. L. Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. In *Proceedings of IEEE Infocom 2000*, Tel-Aviv, Israel, March 2000.

[41] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proc. of ACM Sigmetrics*, 2003.

[42] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to higher moments of conditional response time. In *Proc. of ACM Sigmetrics*, 2005.

[43] A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *Proc. of ACM Sigmetrics*, 2005.

[44] C. W. Yang and S. Shakkottai. Asymptotic evaluation of delay with the srpt scheduler. *To appear in IEEE Transactions on Automatic Control*.

[45] C. W. Yang and S. Shakkottai. Proof for the delay decay rate of finite-srpt. https://webspace.utexas.edu/yangc36/CDC-2007/proof-finite-SRPT.pdf.

[46] C. W. Yang and S. Shakkottai. Delay asymptote of the SRPT scheduler. In *Proceedings of the IEEE Conference on Decision and Control*, December 2004.

[47] C. W. Yang and S. Shakkottai. Asymptotic evaluation of delay with the srpt scheduler. *IEEE Transactions on Automatic Control*, 51(11):1848–1854, November 2006.

[48] C. W. Yang, A. Wierman, S. Shakkottai, and M. Harchol-Balter. Tail asymptotics for policies favoring short jobs in a many-flows regime. In *Proceedings of ACM SIGMETRICS*, June 2006.

[49] S. F. Yashkov. A derivation of responese time distribution for a m/g/1 processor sharing queue. *Problems of Control and Information Theory*, 12(2):133–148, 1983.

[50] A. P. Zwart and O. J. Boxma. Sojourn time asymptotics in the M/G/1 processor-sharing queue. *Queueing Systems*, 35:141–166, 2000.

Vita

ChangWoo Yang was born in Seoul Korea on January 18, 1975, the son of ByungHo Yang and YoungHee Park. After graduating from Karachi American School in 1992, he entered Yonsei University, Seoul, Korea. ChangWoo Yang graduated with a Bachelor's and Master's degree in 1997 and 2001, both in Electrical Engineering from the Yonsei University. He joined the Department of Electrical and Computer Engineering at the University of Texas at Austin in the fall of 2002 as a Ph.D student and is currently working with Prof. Sanjay Shakkottai. While he was in the program, he has awarded the Information and Telecommunication National Fellowship and has worked as a Teaching Assistant for one semester and then a Graduate Research Assistant throughout. In the past, he has worked with LG electronics, Korea Telecom, and the Ministry of Defense.

His research interests lie in the application of probability, queueing theory, and large deviations theory to a variety of communications and computer networking problems.

Permanent Address: 4th floor 498-2 PungNap-Dong SongPa-Gu
Seoul, South Korea

This dissertation was typeset with LaTeX 2ε[3] by the author.

---

[3] LaTeX 2ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.