RANDOMIZATION AND HEAVY TRAFFIC THEORY: NEW APPROACHES TO THE DESIGN AND ANALYSIS OF SWITCH ALGORITHMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Devavrat D. Shah
October 2004

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Balaji S. Prabhakar
(Principal Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ashish Goel

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Rajeev Motwani

Approved for the University Committee on Graduate Studies.

To

my Mother, my Father

and

my Love.

Randomization and Heavy Traffic Theory: New Approaches to the Design and Analysis of Switch Algorithms

By

Devavrat Shah

Abstract

This thesis addresses the design and analysis of implementable high-performance algorithms for high speed data networks, such as the Internet. Our focus is on designing scheduling algorithms for crossbar switches. We exhibit a natural tradeoff between implementational simplicity and goodness of performance for scheduling algorithms operating in very high speed switches. Our goal will be to resolve this tradeoff using novel design methods which involve randomization on the one hand; and to develop new methods to analyze the performance of these algorithms on the other. Along these lines, this thesis has two main parts.

The first part is motivated by the following considerations. The scheduler of a high speed switch poses challenging problems to the algorithm designer. It needs to provide a good performance even though scheduling decisions need to be made in a very limited time and while utilizing meagre computational resources. To illustrate, a switch in the Internet core operates at a line rate of 10 Gbps. This implies that scheduling decisions need to be made roughly every 50 ns. Complicated algorithms cannot be designed to operate at this speed; only the simplest algorithms are implementable. But a simple algorithm may perform rather poorly, if it is not well-designed.

We choose randomization as a central tool to design simple, high-performance switch schedulers. This choice affords us the ability to exploit several desirable features of randomized algorithms: simplicity, good performance, robustness, and the possibility of derandomization for eventual implementation. Specifically, we exhibit three algorithms that exhibit these features.

Our second contribution is a new approach for analyzing the delay induced by a switch scheduling algorithm. Traditional methods, based largely on queueing and large deviation theories, are inadequate for the purpose of analyzing the delays induced by switch schedulers. We adopt a different strategy based on Heavy Traffic Theory which advances our understanding of delay in the following two senses. First, it leads to the characterization of a delay-optimal scheduling algorithm. Second, it helps explain some intriguing observations other researchers have made through simulation-based studies about the delay of scheduling algorithms.

Acknowledgments

My past five years at Stanford have been wonderful. I have many to thank for their generous help and support in making this stay memorable.

First and foremost, I thank my adviser Balaji Prabhakar aka 'Guruji'. I seem to run out of my vocabulary as I try to thank him for all his effort, help, support, guidance and the "Grey hair I caused". His life-lessons and academic wisdom have always amused me. Apart from being a great mentor and a teacher, he has been a wonderful friend to me. I wish all students get an adviser like him!

I am very grateful to Abbas El Gamal for being like my second adviser. I have greatly benefited from his advise. He has influenced my thinking about technical problems to a great extent. I am very thankful to him for being very generous to me.

I am indebted to Rajeev Motwani for being my mentor on the Computer Science side. I thank him for providing opportunities for great research. I have greatly benefited from his advise on all academic aspects. His ways of thinking and attacking problems have influenced me to a great extent.

I appreciate Ashish Goel, Nick McKeown, Rajeev Motwani and Ben Van Roy to take the time and effort from their busy schedule to serve on my defense committee. Ashish and Rajeev are also on my reading committee. I sincerely appreciate their efforts in their help in improving this thesis.

The work of this thesis is done in collaboration. I thank Paolo Giaccone, Milind Kopikare and Damon Wischik for being wonderful collaborator and excellent friends.

Apart from this thesis, I had an opportunity to work with others during my stay at Stanford. Among faculty members, I thank Stephen Boyd and Nick McKeown for their invaluable advise. Among colleagues, I thoroughly enjoyed working with Yahsar Ganjali, Arpita Ghosh, Pankaj Gupta, Sundar Iyer, Neha Kumar, Abtin Keshavarzian, James Mammen, Chandra Nair, Rong Pan and Mayank Sharma.

Many professors at Stanford have strongly influenced me. I thank Mike Harrison for being a great source of inspiration for work on Heavy Traffic Theory (Chapter 4); I thank faculty members on the ISL floor, I thank Persi Diaconis and Amir Dembo in the Stats Department for their wonderful courses, seminars and being sources of inspiration.

I want to thank Denise Murphy for being a wonderful super-efficient and extremely helpful administrator. Her creativity, enthusiasm and (a certain level of) wickedness has made my stay at ISL very enjoyable. A special thanks to Bytes and owner Joe for being around and keeping me awake.

I thank Chandra, Damon (Mosk-Aoyama) and Mayank for their help during the preparation of this thesis.

The "balagroup" has been a enthusiastic, energetic and fun group. I feel privileged to be part of the same group as Abtin, Athina, Chandra, Damon, Elif, Kostas, Mayank, Milind, Mei, Mohsen, Neda, Neha, Rong and Yi. The wonderful parties at Balaji's place, the potlucks at Grad lounges, the movies, the hikes and the great gossips of packard 270 (especially with Chandra, Mayank, Rong and Milind) are some of the unforgettable things for the rest of my life.

I also thank Dhananjay, Kamesh, Muthu, Rohit and Shubha for making my stay on campus a wonderful time. I want to thank my IIT friends Anshul, Avishek, Poojan and Kiran for almost everything.

Finally, I want to thank my family for being very supportive and understanding during my education. I am forever indebted to my late Father, who made sure that I live very well. The values and discipline taught to me by my Mother have proved invaluable for my progress. My sisters, Dhanusha, Lopa and Shesha; Bro-in-Laws Himansu and Shobhit have been very supportive and nephews-niece Rushil, Shailee and Amar have been like a bliss. I especially thank Lopa and Himansu for being wonderful local parents during my stay at Stanford. A special thanks to Dhanusha and Mom for being around while this thesis was written.

Last, but not the least, I thank my love and my wife Salvia for being what she is. I thank her for putting up with me, accommodating my whims, loving me and taking my care. She has

put immense faith in me, provided her help when needed and given me a reason to complete.

Preliminaries

Notation

A very useful representation of state of an $n \times n$ switch is an $n \times n$ real-valued matrix. Hence, a lot of notation used in thesis is matrix based. Let $\mathbb{M}$ be the set of $n \times n$ real-valued matrices, and $\mathbb{M}_+$ the subset consisting of $\mathbb{R}_+$-valued matrices. Let $\mathbb{S}(x)$ be the subset of $\mathbb{M}$ consisting of matrices all of whose row sums and column sums are equal to x, and write $\mathbb{S}$ for $\mathbb{S}(1)$, the set of doubly stochastic matrices. A matrix $\pi = [\pi_{ij}] \in \mathbb{S}$ is called permutation matrix if $\pi_{ij} \in \{0, 1\}$ for all $i,\ j$. Let $\mathbb{P}$ be the set of $n \times n$ permutation matrices. Schedule in a switch will be represented by a permutation matrix, $\pi \in \mathbb{P}$. For a matrix $a \in \mathbb{M}$, write

$$a_{i\cdot} = \sum_j a_{ij}, \quad a_{\cdot j} = \sum_i a_{ij}, \quad a_{\cdot\cdot} = \sum_{i,j} a_{ij}, \text{ and}$$

$$a^* = \max_{i,j}\{a_{i\cdot}, a_{\cdot j}\}, \quad a_* = \min_{i,j}\{a_{ij} : a_{ij} > 0\}.$$

For matrices $a, b \in \mathbb{M}_+$ and function $f : \mathbb{R} \to \mathbb{R}$, let

$$a \cdot b = \sum_{ij} a_{ij} b_{ij},$$
$$ab = \big(a_{ij} b_{ij}\big)_{ij} \in \mathbb{M},$$
$$f(a) = \big(f(a_{ij})\big)_{ij} \in \mathbb{M}.$$

Let component-wise multiplication have precedence over $\cdot$, so that $a \cdot bc = a \cdot (bc)$.

The $\cdot$ operation is commutative. Further, the following distributive law holds.

$$a \cdot (b + c) = a \cdot b + a \cdot c.$$

The well-known Birkhoff-von Neumann's theorem states that the set of all doubly stochastic matrices, $\mathbb{S}$, is a convex set with $\mathbb{P}$ as the set of all possible extreme points. Further, the dimension of the set is $\hat{n} = n^2 - 2n + 1$. Hence, a matrix $a \in \mathbb{S}$ can be written as

$$a = \sum_{k=1}^{\hat{n}} \alpha_k \pi_k,$$

where $\pi_k \in \mathbb{P}$, $\alpha_k \geq 0$ for all k and $\sum_k \alpha_k = 1$. A matrix $b \in \mathbb{M}_+$ is called a doubly sub-stochastic if all of its n row sums and n column sums are no more than 1. A doubly sub-stochastic matrix can be upper bounded component-wise as

$$b \leq b^* a,$$

where $a \in \mathbb{S}$.

Conventions

In this thesis, we assume discrete time packetized network. All packets are assumed to be of the same size. The line-rates are normalized to unit. The packet sizes are chosen so that one packet can arrive in a unit time. In practice, though the packets arriving at a router are of different size, they are internally divided into equal sized "cell"s for the purpose of scheduling.

In an abstract setting, it is possible to consider an $m \times n$, $m \neq n$, switch but in practice each data port of a router acts as an input as well as an output leading to consideration of an $n \times n$ switch. Hence, in this thesis we restrict ourselves to $n \times n$ switch.

We will use the words *schedule*, *matching* and *permutation* interchangeably.

The Maximum Weight Matching scheduling algorithm is central to the study this thesis. Though, many versions of the Maximum Weight Matching algorithm are studied in this thesis depending on the definition of weight function, whenever we write Maximum Weight Matching or MWM without any additional qualifier, we refer to the basic Maximum Weight Matching that uses queue-sizes as weights. See the Section 2.1 for exact definition of the basic MWM algorithm.

How to Read This Thesis

This thesis is about design and analysis of scheduling algorithms for Input Queued switches. The thesis is logically divided into three part: (1) Introduction (Chapter 1), (2) Design methods for switch algorithms (Chapter 3) and (3) Analysis methods for switch algorithms (Chapter 2 and 4). A reader is advised to reach Introduction first. The Design methods and Analysis methods can be read in any order. But, a reader is advised to read Chapter 2 before Chapter 4. Also, if reader decided to read Chapter 3 before Chapter 2, she or he is advised to read statement of Theorem 1 from Chapter 2 for better understanding of motivation for algorithms of Chapter 3.

In Chapters 2-4, we provide references and proper credit to original contributor in the Section titled "Bibliographic Notes" at the end of the chapter. This is done in order to possibly provide a better flow.

This thesis assumes a fair amount of background in Algorithms, Probability theory, Real analysis and Combinatorics. In addition, background in Convex Optimization, Computer Architecture and Router design is useful. Possible reference if required are as follows. For algorithms, a good set of references are *Introduction to Algorithms* by *Cormen et al.* [1990], *Randomized Algorithms* by *Motwani and Raghavan* [1995] and *Data Structures and Network Algorithms* by *Tarjan* [1983]. For Probability theory, some good references are *Probability: Theory and Examples* by *Durrett* [1995] and *Probability and Measure* by *Billingsley* [1995]. In addition, *Brownian Motion and Stochastic Calculus* by *Karatzas and Shreve* [1991] can be useful. For Real analysis and Topology, see *Introduction to Topology and Modern Analysis* by *Simmons* [1963] and *Topology* by *Munkres* [1999]. For Combinatorics, see *A Course in Combinatorics* by *van Lint and Wilson* [1992], *Enumerative Combinatorics* by *Stanley* [1999] and *Combinatorial Algorithms: for computers and calculators* by *Nijenhuis and Wilf* [1978]. For an introductory text on Graph Theory, refer to *Introduction to Graph Theory* by *West* [1996]. For Convex Optimization, refer to *Convex Optimization* by *Boyd and Vandenberghe* [2004] and *Convex Analysis and Optimization* by *Bertsekas et al.* [2003]. For Computer Architecture, see *Computer Architecture* by *Hennesy and Patterson* [1986] and a survey article *Survey on Router Design* by *Keshav and Sharma* [1998] for state-of-art information on router design.

Bibliographic Notes

The preliminary results of Chapter 2 are published as a part of paper by *Shah and Kopikare* [2002]. The results of Chapter 3 are published as part of papers by *Shah et al.* [2001], *Giaccone et al.* [2001], *Giaccone et al.* [2002] and *Giaccone et al.* [2003]. A part of results of Section 4.2, Chapter 4 are published by *Shah* [2001]. The results of the Sections 4.3, 4.4 and 4.5 of Chapter 4 are part of a preprint by *Shah and Wischik*.

Contents

List of Figures

CHAPTER 1

Introduction

The focus of this thesis is the design and analysis of implementable algorithms for problems arising in high speed networks, such as the Internet. Our goals are two-fold: To resolve the tradeoff between implementational simplicity and the goodness of performance of switch scheduling algorithms, to develop new methods for analyzing the performance of these algorithms. Along these lines, this thesis has two main parts.

The first part is motivated by the following considerations. A high speed network presents the algorithm designer with highly constrained problems: the algorithms need to work at a very high speed and utilize limited computational resources, while providing good performance. Consequently, only the simplest algorithms are implementable. But a simple algorithm may perform rather poorly if it is not well-designed. This tension between implementability and high-performance is inherent to the design of crossbar switch scheduling algorithms.

To illustrate this point, let us consider the scheduler of a crossbar switch operating in the core of the Internet. Such switches reside, for example, inside Cisco Systems' GSR 12000 series of Internet routers. The switch operates at a line-rate of 10 Gbps. This implies that the scheduler needs to configure the fabric of the switch roughly once every 50 ns. Each configuration allows the transfer of packets (more precisely, parts of packets) from the inputs to the outputs. This small amount of time and the rather limited computational requirements at a

core router make the design of implementable, high performance schedulers a very challenging problem. The situation will be aggravated in the next generation of routers which will operate at line-rates of 40 Gbps and higher.

Our main approach for designing simple, high-performance switch schedulers is to use randomization. The main idea of randomization is simple to state: Basing decisions on a *small, randomly chosen sample* is a good surrogate for basing decisions upon the complete state. Therefore, randomized algorithms lead to the simple implementation of otherwise complicated solutions. While this general philosophy gives hope, specific problem instances require the designer to exploit the structure of the problem to come up with good randomized algorithms. In this respect we shall see that exploiting the fact that switch scheduling is equivalent to bipartite graph matching is key.

Clearly, the performance of a randomized algorithm depends crucially on the quality of the samples and we are motivated to ask: (a) Is it possible to improve the quality of the samples *without* increasing their *number*? (b) If yes, how well would such an improvement perform? We build on a previous design by *Tassiulas* [1998] to devise a simple trick for recursively improving the sample quality, whilst leaving its size fixed. This trick yields a significant performance boost while retaining the essential simplicity of randomized schemes and has some quite interesting theoretical implications. For example, we shall find that one of our algorithms, Serena, exploits both the structure of matchings and the recursive trick mentioned above to be a very simple, high-performance randomized approximant of the (ideal) maximum weight matching algorithm.

Our second contribution is a new approach for analyzing the delay induced by a switch scheduling algorithm. Traditional methods, based on queueing and large deviation theories for example, are inadequate for the purpose of analyzing delay. We adopt a different strategy based on Heavy Traffic Theory which advances our understanding of delay in the following two senses. First, it leads to the characterization of a delay-optimal scheduling algorithm. Second, it helps explain some intriguing observations other researchers have made through simulation-based studies about the delay of scheduling algorithms.

This thesis is centered around switches that operate in the core of the Internet and which have an Input-Queued (IQ) architecture (for example, GSR 12000 Series Router of *Cisco* [2000]). For the sake of completeness, we will review fundamental concepts from the theory of switching in this chapter. The rest of the chapter is organized as follows. Section 1.1 is devoted to a brief introduction of a typical crossbar switch in the core of the Internet. We describe canonical crossbar-based switch architectures and explain the constraints in building

them. In Section 1.2, we establish the notation that shall be used in the rest of this thesis, and define the problem of scheduling an IQ crossbar switch. We also survey the previous work on scheduling algorithms. In Section 1.3, we discuss in some detail our contributions, and we end with an outline of the rest of the thesis in Section 1.4.

1.1 Switch Architectures

Switching is an integral function of data networks. In an Internet router, packets arrive at various input (ingress) ports destined for any of the output (egress) ports. Figure 1.1 shows the path of a typical packet through a router. On the arrival of a packet at the router, the admission control (AC) module decides whether to admit it or not. Additional policing or pricing mechanisms may be performed at the ingress port. If the packet is admitted, the routing lookup (RL) module decides the output port to which the packet should be sent depending on its final destination and routing information available in locally maintained tables. Subsequently, the packet may be queued before being *switched* to the corresponding output port via the switch fabric. At the output port, the output scheduler (OSch) decides when to transmit the packet on the egress line.

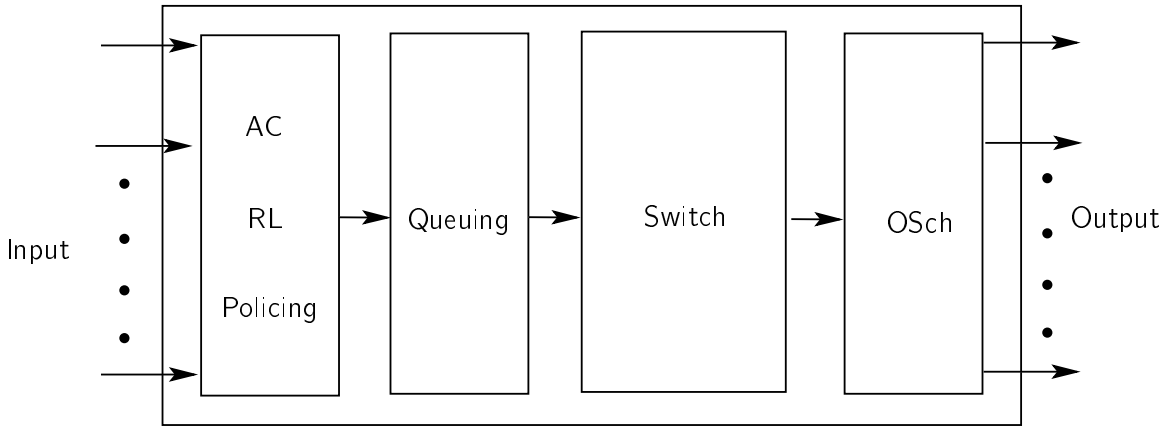

Figure 1.1: Path of a typical packet through a generic Router.

As opposed to the above packet-centric view of a router, Figure 1.2 presents a functional view of the router. The latter representation aggregates modules of the router depending on the kind of information they require to process a packet. Thus, the AC, RL and policing modules require only control information from the header of the packet; whereas the switching

and OSch modules perform data-dependent operations. Note that only the modules in the data plane are affected by the size of packets.

Now, as the speed of a network scales, the router is subject to an increasing amount of computational strain. But whereas the effect on the control plane can be alleviated, say by diving the flow into packets of larger size, problems encountered by the data plane have no such immediate solution. Thus, the functional view clearly identifies those modules that are hit hardest by the scaling of speed of the network, and which need to be addressed effectively. Our work will focus on providing efficient solutions to problems encountered by the data plane modules, in particular, the switch scheduler.

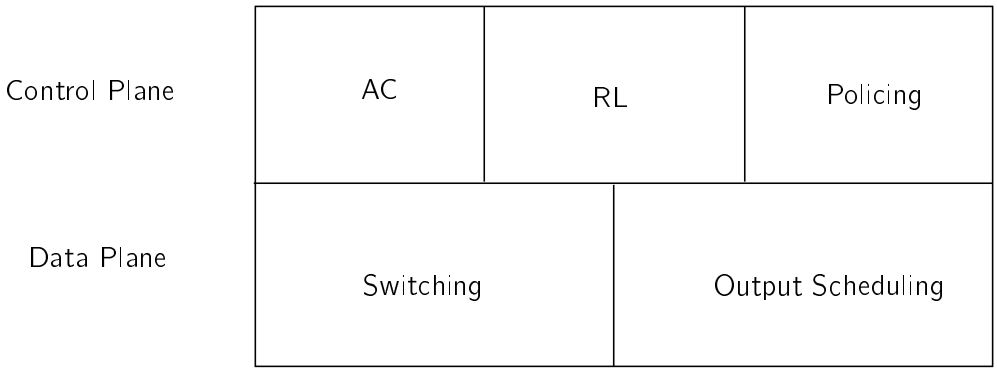

Figure 1.2: Functional view of a Router.

The main function of a switch is to transfer packets from input ports to their destined output ports. An $n \times n$ switch can, by definition, receive packets on n inputs, and is possibly required to send packets out to all n outputs. A schematic diagram of a 3×3 switch is given in Figure 1.3. A switch mainly consists of two parts: (i) Switch fabric, which transfers packets from input to output ports; and (ii) Buffers, which store packets that cannot be sent out immediately. For a switch residing in a core router, line-rates are on the order of Gbps. For example, in the current OC-192 standard, the line-rate is 10 Gbps. Soon, the line-rate is expected to increase to 40 Gbps when OC-768 standard is adopted. The buffer of a switch must operate at a rate that is at least twice the line-rate (corresponding to a read and a write operation per time slot).

In recent years, due to the rapid and ubiquitous deployment of optical fibers, the line-rate has increased at a very fast pace. Roughly speaking, line-rates have doubled every 12 months. This should be contrasted with the fact that memory speed is doubling every 18

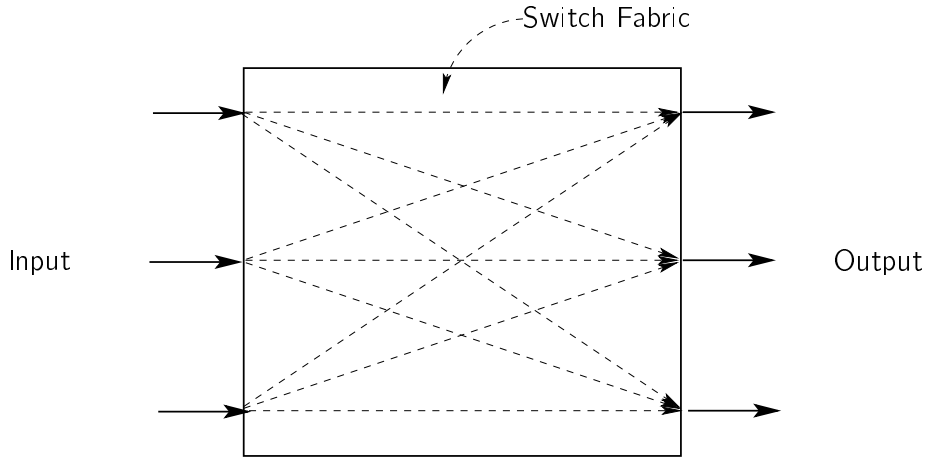

Figure 1.3: A Schematic Diagram of a 3×3 Switch.

months according to Moore's law *Hennesy and Patterson* [1986]. Currently it is barely feasible to build a switch with memory fast enough to operate at the line-rate. In the future, it is likely to become extremely challenging to build switches that operate at line-rate. (The source of the above information is *McKeown*).

Given that memory bandwidth is one of the most significant constraints in building high speed switches, in what follows, the relative goodness of a switch architecture shall be decided by its memory-bandwidth requirement.

Next we discuss three popular switch architectures, which differ essentially in the placement of buffers.

1. Output-Queued (OQ) switches, where buffers are at the output port,
2. Input-Queue (IQ) switches, where buffers are at the input port, and
3. Combined Input-Output Queued (CIOQ) switches, where buffers are at both the input and the output port.

1.1.1 Output-Queued Switch

Figure 1.4 shows a 3×3 OQ switch. In an OQ switch, arriving packets are directly transfered from input to output ports and stored in the buffers residing at the output ports, if required. In such a switch, only the packets destined for the same output will contend for sharing bandwidth of the outgoing line. This is the least contention of bandwidth expected

in any switch. This makes an OQ switch an ideal switch in terms of performance. But an OQ switch requires huge memory bandwidth: in an $n \times n$ OQ switch, the buffer memory is required to run $n+1$ times faster than the line-rate because possibly n packets arrive and one packet departs from the same output port in a time slot. As discussed above, the limitation on memory bandwidth makes it infeasible to build high-speed OQ switches with large number of ports.

Though unbuildable, the performance of the OQ switch is ideal. Hence, it is used as a theoretical reference to which the performance of other switches can be compared. A detailed exposition on this topic can be obtained in the works by *Prabhakar and McKeown* [1999], *Chuang et al.* [1999], *Iyer et al.* [2002], *Iyer* [2002], *Keslassy* [2004], *Shah* [2003], *Krishna et al.* [1999] etc.

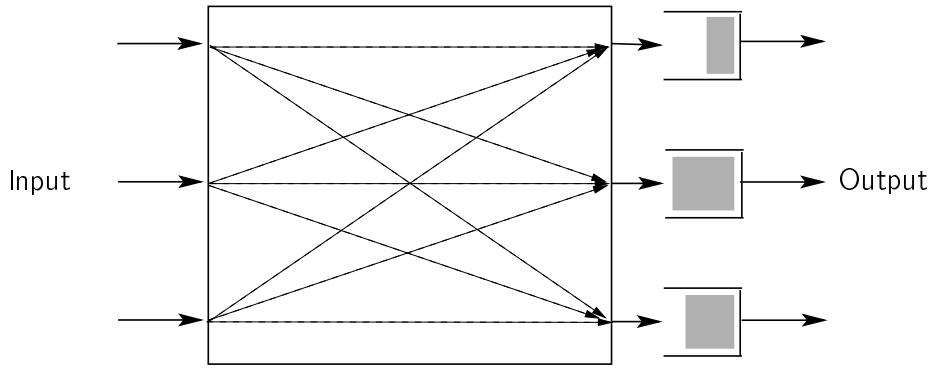

Figure 1.4: An example of an Output-Queued Switch.

1.1.2 Input-Queued Switch

Figure 1.5 shows a 3×3 IQ switch with a crossbar switch fabric. The arriving packets are stored in the buffers at the input side. At each input, there are separate buffers for each output, which are called Virtual Output Queues (VOQ). The crossbar fabric imposes the following logical constraints: in a time slot, each input can transfer at most one packet to any output and each output can receive at most one packet from an input. For example, Figure 1.5 shows an instance when input 1 is connected to output 1, input 2 to output 2 and input 3 to output 3. Due to the crossbar fabric, at most one packet arrive at each output port in a time slot. Hence, buffers are not needed at the output ports.

The crossbar constraints require the buffer memory to run only twice (one for read and

one for write) the line-rate of a switch of any number of ports. This low memory bandwidth requirement makes it possible for an IQ switch to operate at very high speed. Though crossbar constraints are useful for low memory bandwidth, they create the following scheduling problem: in every time slot a scheduling algorithm is required to find a "schedule" of the packets which form a "matching" between inputs and outputs. Now, the performance of a switch depends on the scheduling algorithm. For good performance, the algorithm is required to find a good schedule. Further, engineering constraints require it to be simple so as to be implementable. In this thesis, we present methods for designing implementable high performance scheduling algorithms.

The IQ switch architecture has been studied for more than a decade. It was first introduced by *Karol et al.* [1987]. Later, the works of *Tamir and Chi* [1993]*Anderson et al.* [1993]*Karol et al.* [1992] led to the development of the theory of switch scheduling. The Section 1.2 introduces the problem of scheduling formally.

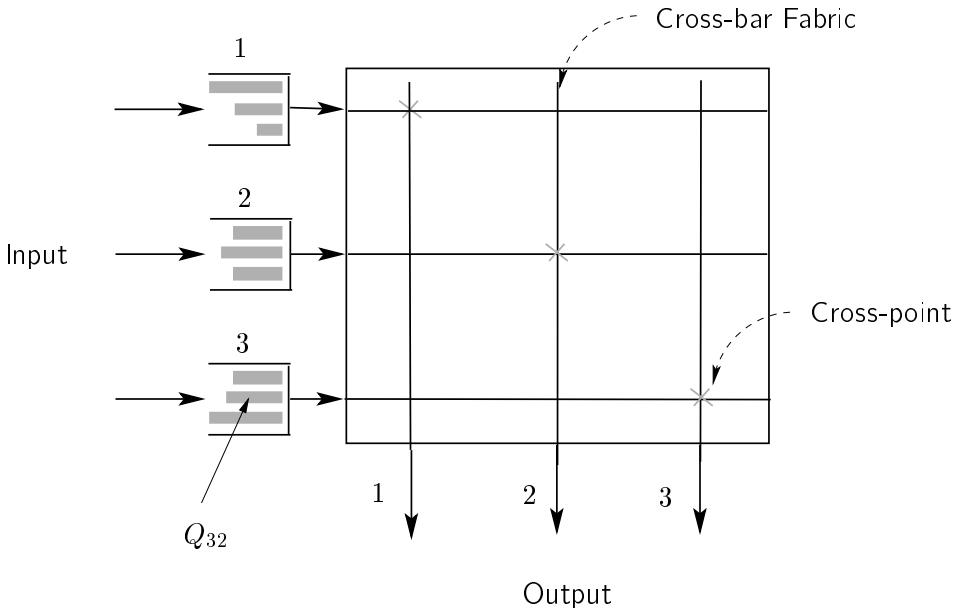

Figure 1.5: An example of an Input-Queued Switch.

1.1.3 Combined Input-Output Queued Switch

Figure 1.6 shows a 3×3 CIOQ switch with a crossbar fabric. A CIOQ switch is essentially an IQ switch with the crossbar fabric running at a rate higher than the line-rate. If the crossbar

fabric runs s times faster than the line-rate, then the CIOQ switch is said to have *speedup* s. The speedup $s > 1$ in a CIOQ switch requires it to have buffers at both input and output ports. The buffers are required to operate rate $s+1$ times the line-rate in a CIOQ switch with speedup s.

The CIOQ switch architecture was formally introduced by *Prabhakar and McKeown* [1999]. They showed the possibility of emulating the performance of an OQ switch by a CIOQ switch with a constant* speedup. However, the algorithms required for this emulation are very complex to implement due to the communication overhead in a computing schedule and the requirement of additional speedup.

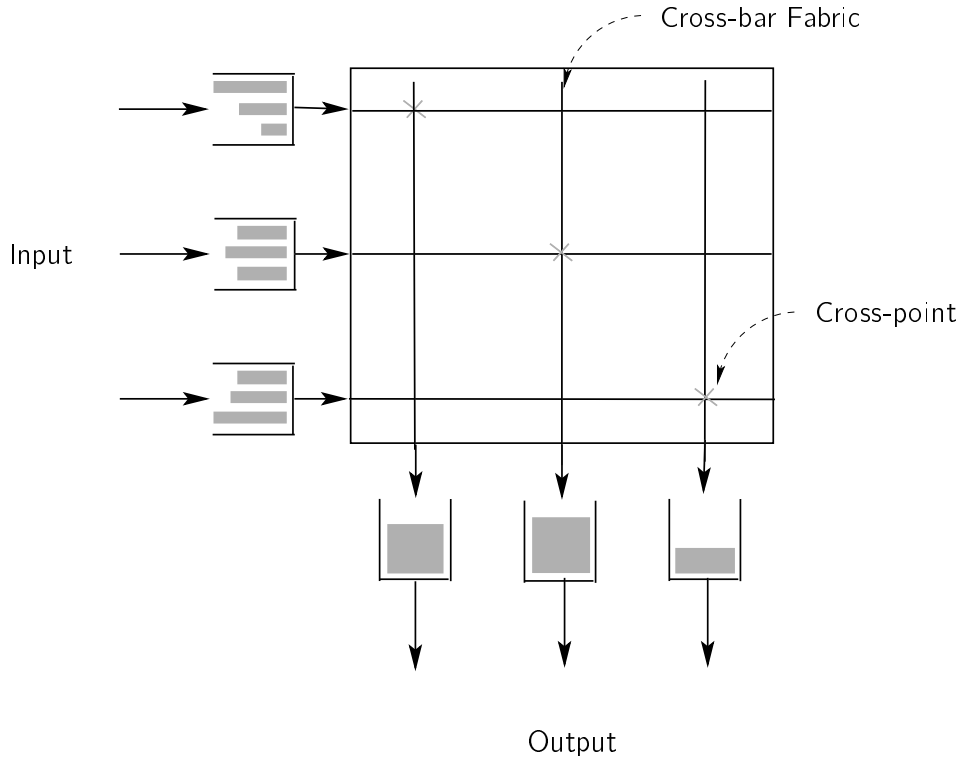

Figure 1.6: An example of a Combined Input-Output Switch.

*Speedup 4 was shown to be sufficient in *Prabhakar and McKeown* [1999]. In *Chuang et al.* [1999] speedup 2 was shown to be necessary and sufficient.

1.2 Scheduling in IQ Switch

Consider an $n \times n$ IQ switch. The packets arriving at input i destined for output j are stored in VOQ (i,j). The occupancy of VOQ (i,j) is represented by Q_{ij}. As noted before, the crossbar fabric imposes the following constraints: in a time slot, (i) each input can transfer at most one packet, and (ii) each output can receive at most one packet. The switch scheduling problem is to find a schedule of packets satisfying the above constraints.

A natural and a very useful representation of an IQ switch is a weighted bipartite graph. A weighted bipartite graph corresponding to a 3×3 IQ switch is shown in the Figure 1.7(a). The nodes on the left represent inputs and the nodes on the right represent outputs. An edge between input i and output j corresponds to (a non-empty) queue (i,j). Edge (i,j) is assigned weight which is a function of the state of the switch. For example, weight of the edge (i,j) can be queue-size Q_{ij} or a function of Q_{ij}. A matching[†] in such a weighted bipartite graph corresponds to a possible schedule in the IQ switch. The Figure 1.7(b) shows one of the possible matchings or schedules for the bipartite graph in part (a) of the figure. Thus, a scheduling algorithm is equivalent to a matching algorithm on a weighted bipartite graph.

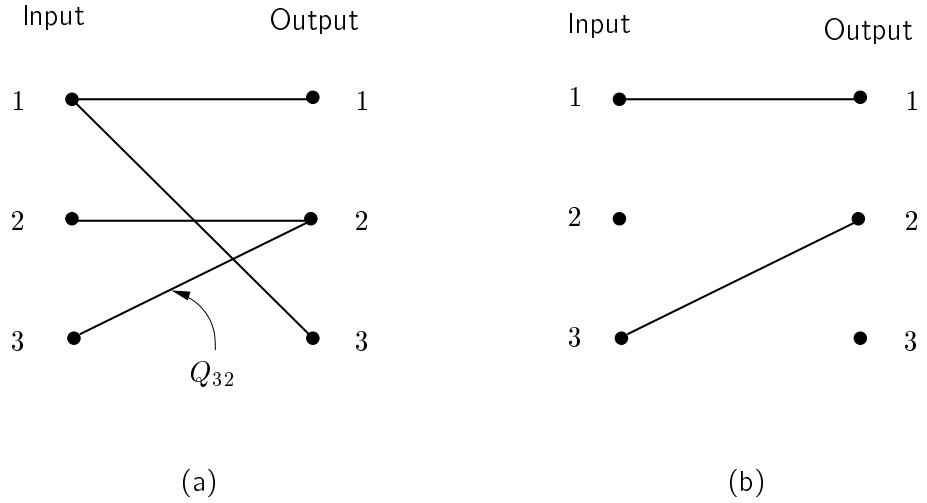

Figure 1.7: (a) Bipartite graph corresponding to a 3×3 switch. (b) A matching corresponding to a valid schedule.

In the rest of the section, we introduce notation, definitions and the switch dynamics that shall be used in this thesis.

[†]A matching is a collection of edges such that no two edges are incident on the same vetrex.

1.2.1 Notation, Setup, and Dynamics of a Switch

Let time be indexed by m. Initially, $m = 0$. Let the $n \times n$ integer valued matrix $Q(m) = [Q_{ij}(m)]$ denote the queue-sizes of the switch at time $m \geq 0$. We assume that the switch starts empty, i.e. $Q(0) = [0]$. For reasons that will become apparent later in the thesis, we call $Q_{i\cdot}(m)$ the workload at input i; $Q_{\cdot j}(m)$ the workload at output j at time m; and $Q_{\cdot\cdot}(m)$ the overall workload in the switch. We are interested in the dynamics of $Q(\cdot)$, which depends on the arrival and service process. The arrival process is exogenous while the service process depends on the scheduling algorithm. Next, we describe the necessary notation and assumptions on the arrival and service processes.

Let $\bar{A}(m) = [\bar{A}_{ij}(m)]$ denote the cumulative arrival process until time m, i.e. $\bar{A}_{ij}(m)$ denote the number of packets arrived at input i for output j in the time interval $[0, m]$. Let $A_{ij}(m) = \bar{A}_{ij}(m) - \bar{A}_{ij}(m-1)$ be the number of packets arriving at input i for output j in time slot m. Since at most one packet can arrive at input i in a time slot, the $A_{ij}(m)$ are 0-1 variables. Let $u_{ij}(k)$ denote the inter-arrival time between the $(k-1)^{st}$ and k^{th} packet at input i for output j. Thus,

$$\bar{A}_{ij}(m) = \max\{\ell \ : \ \sum_{k=1}^{\ell} u_{ij}(k) \leq m\}.$$

Similarly, $\bar{D}(m) = [\bar{D}_{ij}(m)]$ denotes the cumulative departure process from $Q(m)$, and $D(m)$ denoted the number of departures in time m. We assume that $\bar{A}(0) = \bar{D}(0) = [0]$.

Now, the line-rates are normalized to one, and hence at most one packet can arrive at an input and at most one packet can depart from an output in a given time slot; i.e. for all $m, \ell \geq 0$ and for all i, j,

$$\bar{A}_{ij}(m+\ell) - \bar{A}_{ij}(m) \leq \ell, \qquad \bar{D}_{ij}(m+\ell) - \bar{D}_{ij}(m) \leq \ell. \tag{1.1}$$

Additionally, we assume that the arrival process satisfies the following assumption.

Assumption 1. *The inter-arrival times $(u_{ij}(\cdot))$ are IID random variables for all i, j. Let the arrival rate-matrix be $\lambda = [\lambda_{ij}]$, that is,*

$$E[\bar{A}(1)] \ = \ \lambda. \tag{1.2}$$

Further, whenever $\lambda_{ij} \neq 0$ (i.e. packets arrive at input i for output j),

$$E[u_{ij}^2(1)] \quad < \quad \infty. \tag{1.3}$$

Under a Bernoulli IID arrival process, $\{A_{ij}(m), m \geq 1\}$ are Bernoulli IID random variables with $\Pr(A_{ij}(1) = 1) = \lambda_{ij}$. Note that, the Bernoulli IID arrival process satisfies Assumption 1 as stated above. As we shall see later in the thesis, the Bernoulli IID arrival process provides us with a good understanding of the throughput and delay under various switch algorithms.

Assumption 2. *We assume that the switch starts empty at time 0, that is,*

$$Q(0) = [0]. \tag{1.4}$$

The line-rates are one and hence by definition $\lambda_{i\cdot}$ is at most 1 for all i. Since the output line-rates are one, in order to have finite queue sizes, $\lambda_{\cdot j}$ is required to be less than 1. Motivated by this, we call an arrival rate-matrix λ as *admissible* if it is strictly doubly sub-stochastic, i.e.

$$\lambda_{i\cdot} < 1; \qquad \lambda_{\cdot j} < 1, \quad \forall\ i,\ j. \tag{1.5}$$

We say that an input (output) port i (j) is *critically loaded* if $\lambda_{i\cdot} = 1$ ($\lambda_{\cdot j} = 1$).

In switches, queues are served by schedules (or permutations). Hence, the service process (subsequently departure process) is completely determined by $\{S_\pi(m), \pi \in \mathbb{P}, m \geq 0\}$, where $S_\pi(m)$ denotes the cumulative amount of time a scheduling algorithm chooses to serve permutation π in the time interval $[0, m]$. Let $S_\pi(0) = 0,\ \forall \pi \in \mathbb{P}$.

Now, we are ready to describe the dynamics of the switch. The dynamics of a switch have two components: (1) Algorithm-independent dynamics, and (2) Algorithm-dependent dynamics.

Algorithm-independent dynamics

The dynamics of a switch are completely described by the quantities $Q(\cdot), A(\cdot), D(\cdot)$ and $S(\cdot) = (S_\pi(\cdot))_{\pi \in \mathbb{P}}$. That is, the tuple $\mathcal{X}(\cdot) = (Q(\cdot), A(\cdot), D(\cdot), S(\cdot))$ describes the switch. These quantities are related by the following basic queueing equation.

$$\begin{aligned} Q(m) &= Q(0) + \bar{A}(m) - \bar{D}(m) \\ &= \bar{A}(m) - \bar{D}(m), \end{aligned} \tag{1.6}$$

since $Q(0) = 0$ from Assumption 2. In each time slot, at most one of the permutations is served, and we are interested in non-idling switches. Hence,

$$\sum_{\pi \in \mathbb{P}} S_\pi(m) \quad = \quad m. \tag{1.7}$$

Clearly, $\bar{D}(m)$ and $\{S_\pi(\cdot),\ \pi \in \mathbb{P}\}$ are related to each other. Specifically,

$$\bar{D}_{ij}(m) \quad = \quad \sum_{\pi \in \mathbb{P}} \sum_{\ell=1}^{m} \pi_{ij} \mathbf{1}_{Q_{ij}(\ell)>0} \left(S_\pi(\ell) - S_\pi(\ell-1)\right), \quad \forall\ i,\ j. \tag{1.8}$$

Equivalently,

$$\bar{D}_{ij}(m) - \bar{D}_{ij}(m-1) \quad = \quad \sum_{\pi \in \mathbb{P}} \pi_{ij} \mathbf{1}_{Q_{ij}(m)>0} \left(S_\pi(m) - S_\pi(m-1)\right), \quad \forall\ i,\ j. \tag{1.9}$$

Note that, the equations (1.6)-(1.9) hold for a switch with any scheduling algorithm.

Algorithm-dependent dynamics

Now we describe the dynamics of a switch that depends on the algorithm, unlike the above equations. In particular, an algorithm decides which permutations are chosen for service, that is, $\{S_\pi(\cdot), \pi \in \mathbb{P}\}$. Here, we describe the dynamics for the following algorithms of particular interest: (1) a very well-studied algorithm called the Maximum Weight Matching algorithm, (2) Maximum Size Matching, (3) Maximal Matching, and (4) Round-Robin algorithm.

(1)*Maximum Weight Matching*. Consider the switch bipartite graph, as in Figure 1.7. Let the edge (i,j) be assigned weight $Q_{ij}(m)$ at time m. Then, the basic Maximum Weight Matching algorithm, denoted by MWM, selects a schedule corresponding to the maximum weight matching in the bipartite graph. Equivalently, at time m, MWM chooses a permutation, $\pi^*(m)$ such that

$$\pi^*(m) \quad = \quad \arg\max_{\pi \in \mathbb{P}} \pi \cdot Q(m). \tag{1.10}$$

An equivalent condition is the following.

$$S_\pi(m) = S_\pi(m-1) \quad \text{if} \quad \pi \cdot Q(m) < \max_{\rho \in \mathbb{P}} \rho \cdot Q(m), \quad m \in \mathcal{Z}_+. \tag{1.11}$$

Now, if edge (i,j) is given weight $f(Q_{ij}(m))$ for some function $f : \mathbb{R}_+ \to \mathbb{R}_+$, then the corresponding Maximum Weight Matching, denote by MWMf, satisfies the following conditions:

$$S_\pi(m) = S_\pi(m-1) \quad \text{if} \quad \pi \cdot f(Q(m)) < \max_{\rho \in \mathbb{P}} \rho \cdot f(Q(m)), \quad m \in \mathcal{Z}_+. \tag{1.12}$$

An MWMf algorithm using $f(x) = x^\alpha,\ \alpha \in \mathbb{R}_+$, is denoted by MWM-$\alpha$. In this thesis, we will study the properties of MWM-α algorithms in a great detail.

(2) *Maximum Size Matching*. The Maximum Weight Matching algorithm assigns the queue size (or a function of it) as the weight of an edge, and serves the maximum weight matching. Instead, consider the following weight: let the weight be 0 if the queue is empty and 1 otherwise. The Maximum Weight Matching algorithm with respect to this weight serves the matching that maximizes the number of packets transferred. That is, the algorithm serves a maximum size matching. The Maximum Size Matching (MSM) algorithm satisfies the following equations.

$$S_\pi(m) = S_\pi(m-1) \quad \text{if} \quad \pi \cdot \mathbf{1}(Q(m)) < \max_{\rho \in \mathbb{P}} \rho \cdot \mathbf{1}(Q(m)), \quad m \in \mathcal{Z}_+, \tag{1.13}$$

where the function $\mathbf{1}(x) = \begin{cases} 1, & \text{if} \quad x > 0, \\ 0, & \text{otherwise.} \end{cases}$

(3) *Maximal Matching*. The use of a word maximal matching is not unique to one particular algorithm, but is a characteristic of a large class, including MWM and MSM described above. Intuitively, an algorithm is called maximal if the schedule used by algorithm is such that no more packets can be transferred in the same time slot, in addition to the packets transferred by the algorithm, while obeying the matching constraints. Precisely, a Maximal Matching algorithm satisfies the following conditions.

$$Q_{ij}(m) > 0 \Rightarrow \left[\sum_{\pi \in \mathbb{P}} \sum_{k=1}^{n} (S_\pi(m) - S_\pi(m-1))(\pi_{ik} \mathbf{1}_{Q_{ik}(m)>0} + \pi_{kj} \mathbf{1}_{Q_{kj}(m)>0}) \right] > 0 \tag{1.14}$$

(4) *Round-Robin*. Let all $n!$ permutations of $\mathbb{P}$ be numbered from $1, \ldots, n!$ in some order. Let $\pi(l)$ denote the permutation numbered l according to this order. The Round-Robin (RR) algorithm selects the schedule corresponding to the permutation $\pi(m \mod n! + 1)$ at time m. Hence, under the RR algorithm, the switch obeys the following equations.

$$S_{\pi(l)}(m) = S_{\pi(l)}(m-1) + \mathbf{1}_{\{l=(m \mod n!+1)\}}, \quad m \in \mathcal{Z}_+. \tag{1.15}$$

1.2.2 Performance Measures

The performance of a scheduling algorithm is measured in terms of throughput and average packet delay. Intuitively, throughput is the rate at which the switch can transfer data from inputs to outputs. As discussed above, any rate λ that can be transferred by switch has to be admissible. Next, we define the notion of stability or 100% throughput.

Definition 1 (Stable Algorithm). *A scheduling algorithm is called rate-stable (equivalently, it is said to deliver 100% throughput) if under any arrival process satisfying Assumption 1 and admissible rate-matrix λ, the departure process is such that*

$$\lim_{m\to\infty} \frac{D(m)}{m} = \lambda, \quad \textit{with probability } 1.$$

A rate-stable algorithm is called strongly stable if

$$\limsup_{m\to\infty} E[Q_{ij}(m)] < \infty, \ \forall i, j.$$

The delay of a packet is the time spent by the packet in the switch until it departs. By Little's Law, average delay is related to average queue-size for a stable system. Hence, in this thesis, we may use the words average delay and average queue-size interchangeably.

1.2.3 Previous work on Scheduling Algorithms

Input-Queued switch scheduling algorithms have been very well studied in the last decade or so. A lot of research has been done by people in industry and academics to obtain implementable scheduling algorithms with good performance guarantees. To evaluate the performance of scheduling algorithms, a great deal of theory has been developed. Unfortunately, not much success has been achieved either in terms of designing good implementable algorithms or in developing theory to analyze the delay of scheduling algorithms.

Previous work on Design of Algorithms

The initial work on the design of scheduling algorithms focused on obtaining stable scheduling algorithms. *McKeown et al.* [1996] showed that under a Bernoulli IID arrival process, Maximum Weight Matching (with queue-size as weight) is stable. A similar result in the context of Radio-hop networks was obtained by *Tassiulas and Ephremides* [1992]. Recent results by

Prabhakar and McKeown [1999], *Chuang et al.* [1999] and *Krishna et al.* [1999] proposed algorithms for CIOQ switches to emulate the performance of an OQ switch with speedup between 2 and 4. These algorithms are stable and permit the use of sophisticated mechanisms for supporting quality-of-service (QoS).

However, the above algorithms are too complicated to implement. For example, the best known algorithm to find a Maximum Weight Matching requires $O(n^3)$ operations in the worst case *Edmonds and Karp* [1972]. That is, for a 30-port switch, it will require 27000 operations. Thus, a switch operating at 10Gbps, with packet size of 50 bytes, will be required to do this many operations roughly every 5-10ns. This is infeasible under current technology. Further, due to the back-tracking nature of the routine involved in such an algorithm, it is not suitable for pipelining. Similar reasons hold for other well-known algorithms.

Implementation considerations have therefore led to the proposal of a number of practicable scheduling algorithms. A very successful algorithm, called iSLIP, was proposed by *McKeown* [1995] and *McKeown* [1999]. The iSLIP algorithm is a maximal matching algorithm with the possibility of distributed implementation. Due to the simplicity of iSLIP, its variants are implemented in some commercially available routers. The iSLIP algorithm, though very simple to implement, performs poorly. To improve the performance while retaining simplicity a number of other algorithms have been proposed; notably iLQF by *McKeown* [1995], RPA by *Marsan et al.* [1999], MUCS by *H.Duan et al.* [1997], Parallel Iterative Matching by *Anderson et al.* [1993] and Wave Front Arbiter by *Tamir and Chi* [1993]. However, these algorithms perform poorly compared to MWM when the input traffic is non-uniform: they induce very large delays and their throughput can be less than 100%.

More recently, some particularly simple-to-implement scheduling algorithms have been proposed by *Chang et al.* [2001] and by *Iyer* [2002] and proven to be stable. But these algorithms require multiple switch fabrics. Essentially they reduce the complexity of the scheduling algorithm by additional (expensive) resources. Nevertheless, these algorithms demonstrate a significant point: delivering 100% throughput does not complicate the scheduling problem. On the other hand, in order to keep delays small, it seems necessary to find very good matchings; and finding good matchings is generally very hard, requiring complex algorithms.

Previous work on Analysis of Algorithms

A significant amount of research has been done to develop methods for analyzing the performance of algorithms. A great amount of success has been achieved in developing methods

for throughput analysis, but delay analysis methods are still lacking.

Throughput analysis methods are mainly based on Lyapunov function theory and fluid model techniques. The method of using Lyapunov functions is quite ancient. In the context of switching, it was first used by *Tassiulas and Ephremides* [1992] and *McKeown et al.* [1996] to prove the stability of the Maximum Weight Matching algorithm under Bernoulli IID arrival processes. Subsequently, it has been utilized very heavily. For example, in *Tassiulas* [1998]*Keslassy and McKeown* [2001a]*Giaccone et al.* [2003]*Marsan et al.* [2003].

The fluid model technique is one of the significant development of the 1990s for throughput analysis of stochastic networks. *Dai and Prabhakar* [2000] were the first ones to apply the fluid model technique in the context of switch scheduling algorithms. They proved rate-stability of the MWM algorithm and showed that any maximal matching is stable for a CIOQ switch at speedup 2 or more. This method is quite general and has been used extensively. For example, in *Shah* [2001].

The definition of throughput assumes availability of infinite size buffers. In practice, routers have finite size buffers. Hence, sometimes throughput fails to capture the notion of "practical capacity". To explain this, we present an example. Consider two algorithms, Algo2 and iSLIP. A detailed description of the Algo2 can be found in Section 3.1 of Chapter 3. The Algo2 provides 100% throughput (i.e. stable) while iSLIP algorithm is believed to be unstable for non-uniform traffic. Now for a particular non-uniform traffic pattern (called Diagonal traffic pattern), we find the simulation results as shown in Figure1.8. The Figure 1.8 plots average queue-length versus the normalized load for various algorithms. The performance under the MWM algorithm is plotted as a reference. The figure suggests that at load 0.5 (i.e. at 50% loading) the performance of iSLIP is vastly better than Algo2. In particular, at the load of 0.5 the average queue-size under iSLIP is less than 10 while the average queue-size under Algo2 is so large that can not be plotted in the figure (i.e. a lot larger than 10000). Thus, if a router has buffer size equal to 1000, then the effective throughput achieved at load 0.5 under iSLIP is at least is 99% of arriving traffic while the Algo2 will certainly lose a significant fraction of the throughput. Thus, iSLIP seems much better algorithm than Algo2 for particular situation explained above.

The above example motivates the necessity of studying queue-size or delay induced by an algorithm. Unlike throughput analysis methods, delay analysis methods are not well developed. The main reason is the inherent difficulty in analyzing delay in complex systems like switches. However, some interesting approaches for analysing the delay of a switch algorithm have been

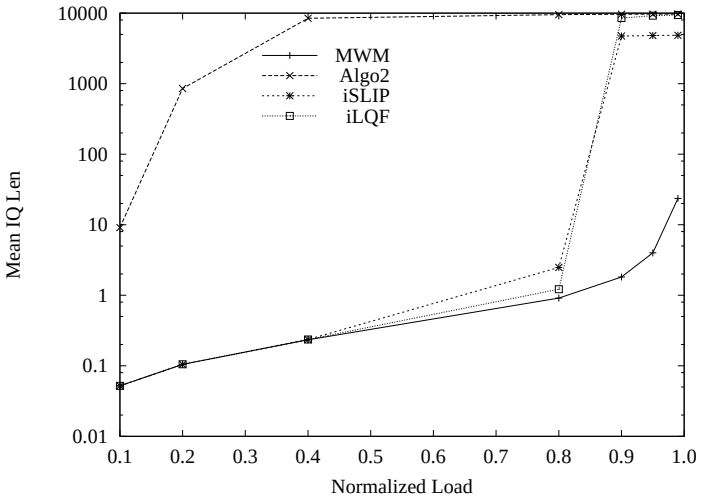

Figure 1.8: Comparison of Algo2 and iSLIP.

developed, which we now describe.

Prabhakar and McKeown [1999] introduced the notion of Output Queued switch emulation. This allows for the evaluation of the delay of an algorithm as it is relatively easy to explicitly evaluate delay of an OQ switch for a large class of arrival process. Unfortunately, OQ emulation is a rare property and hence this strategy is not very useful in general.

Leonardi et al. [2001] obtained delay bounds for the Maximum Weight Matching algorithm for Bernoulli IID arrival process. Unfortunately, their method, as presented, does not seem to apply well to general algorithms.

Summary of Previous Work

The previous work can be summarized with the help of Figure 1.9. This figure plots the known algorithms and architectures with respect to implementability and performance. The OQ switch as well as the CIOQ switch (emulating an OQ switch) are ideal in performance but practically infeasible to implement in a high speed switch. The iSLIP algorithm and its variants are very good in terms of implementation but very poor in performance. Algorithms based on Maximum Weight Matching provide Statical guarantees but still remain unimplementable. The questions that remain open are: (i) what is an implementable algorithm that is good in performance? (A in Figure 1.9); and (ii) what is an ideal scheduling algorithm for an IQ switch

in terms of throughput and delay? (B in Figure 1.9).

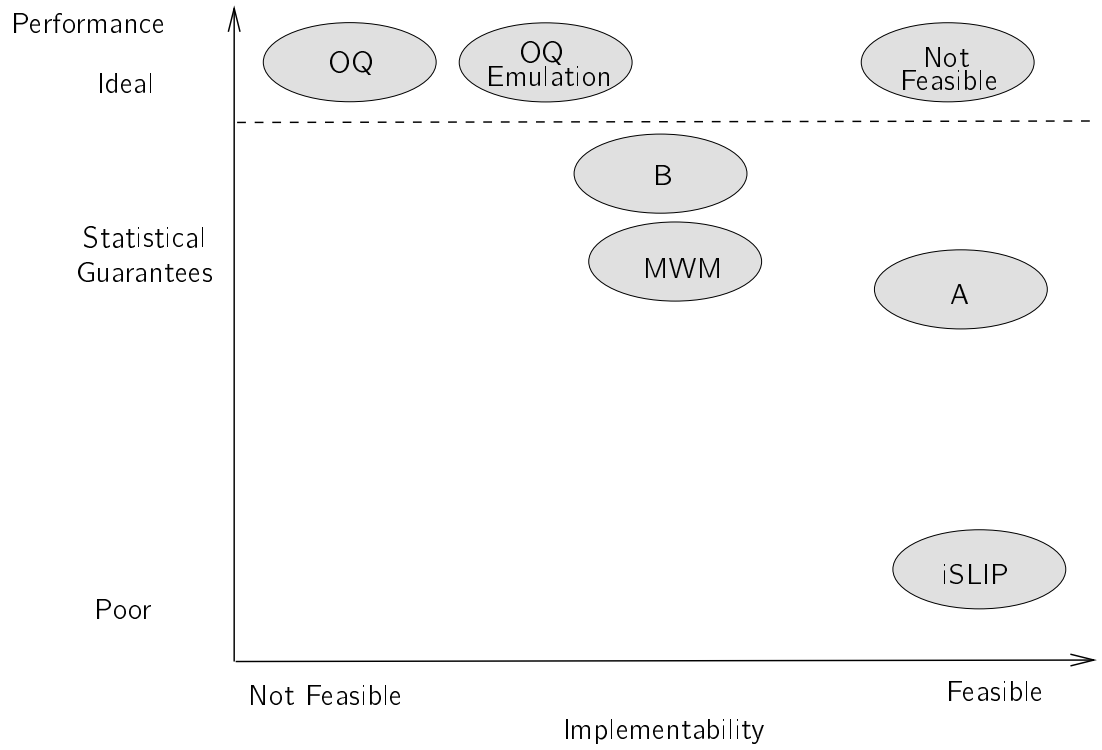

Figure 1.9: Summary of previous results: performance v/s implementability.

1.3 Contributions

This thesis has two main contributions: first, we develop new design methods for implementable algorithms with performance guarantees; second, we develop a new analysis method, based on Heavy Traffic theory, to study the delay of algorithms.

1.3.1 Design Methods

We exploit three design techniques to obtain simple-to-implement high performance scheduling algorithms: (1) Randomization with Memory, (2) Use of arrival information and (3) Parallelism.

For more than a decade, randomization has been used in many problems to design simple algorithms (see *Motwani and Raghavan* [1995]). The basic idea behind randomized algorithms

is as follows: *the decision is based on a few randomly chosen samples instead of the whole state*. In many applications, clever choice of few random samples gives excellent performance. Unfortunately, in the context of switch scheduling, randomization alone does not help in obtaining a good scheduling algorithm. Observe though that the state (i.e. queue-sizes) of a switch changes very little between successive time slots. Hence a heavy schedule remains heavy (with respect to queue-size as weight) in the successive time slots. Thus, the use of information from the past, or memory, is very useful. We use randomization and memory along with the structure of matchings to obtain the high performance algorithm LAURA.

In switches, the goal of a scheduling algorithm is to keep the delay or queue-sizes small. To do so, the algorithm should serve longer queues with higher priority. The queues to which arrivals happen often are more likely to be longer. Hence, looking at the queues that are *exposed* by arrivals leads to a way to discover good schedules. The algorithm SERENA is based on this idea.

Finally, the structure of permutations allows for the parallelism in discovering good schedule from a previous schedule. We use this idea to obtain the algorithm APSARA.

The algorithms implemented in the current routers (for example *Cisco* [2000]) have poor performance. Hence, in order to guarantee high performance, an ISP over provisions the network in terms of routers. We believe that by employing the algorithms proposed in this thesis (especially APSARA and SERENA), the performance of routers will improve significantly. Hence, the ISP using these new routers will require a lot fewer routers in order to guarantee the same level of performance. Consequently, the cost of operating a core-network will reduce drastically.

1.3.2 Analysis Methods

Perhaps the most important contribution of this thesis is the delay analysis method based on the Heavy Traffic Theory.

The Heavy Traffic theory has been well developed over the past two to three decades. As the name suggests, roughly speaking under heavy traffic scaling the system is loaded critically. In this regime, for many networking systems, a phenomenon called "state space collapse" occurs. This means that the state of the system under heavy traffic lives in a smaller dimensional space compared to the original space. *Stolyar* [2004] studied the state space collapse property of MWM algorithms under the special case of heavy traffic in which only one logical resource (i.e. one input port or one output port) is saturated while the rest

are underloaded. The result obtained by *Stolyar* [2004] strongly depends on the fact that only one logical resource is saturated. The techniques do not extend to the case when multiple resources are saturated.

In this thesis, we study the switches under heavy traffic when one or more ports are saturated. When all ports are saturated, we find that the state space collapse region is different for different algorithms, unlike the results of *Stolyar* [2004], who finds the same state space collapse for all algorithms. Our results build on the recent work by *Bramson* [1998] and *Williams* [1998] in the heavy traffic theory.

The state space collapse characterization of algorithms extends our understanding of the performance of algorithms. First, we use this characterization to find an optimal algorithm in terms of throughput and average delay. In particular, we show that the formal limit of MWM-α algorithm as $\alpha \to 0^+$ is an optimal algorithm. As explained in Chapter 4, this is a Maximum Size Matching algorithm which breaks ties among multiple maximum size matching by selecting the maximum weighted maximum size matching.

Next, we use this technique to demonstrate that the MWM (i.e. MWM-1) algorithm is not optimal. Thus, we show that the long-standing folk-lore in the switching community about the optimality of MWM is false.

Finally, we use these results to explain the following intriguing conjecture made by *Keslassy and McKeown* [2001a] based on empirical observations.

Conjecture 1. *For $\alpha \in \mathbb{R}_+$, the average delay of the MWM-α algorithm decreases as α decreases.*

It can be shown that all MWM-α algorithms are stable for $\alpha \in \mathbb{R}_+$, using the traditional method based on fluid model (see Section 4.2 of Chapter 4). But, traditional methods for delay analysis are not useful in explaining the delay behavior of algorithms as claimed by the Conjecture 1. Again, we use the state space collapse characterization of MWM-α algorithms to explain the observed monotonicity in the delay behavior of the MWM-α algorithms.

Our methods are general and we believe that they can be easily extended to other scheduling problems where a scheduling decision corresponds to an extreme point of a closed and bounded convex set in $\mathbb{R}^d$, for some finite d.

1.4 Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we prove the throughput and delay properties of MWM and its approximations under Bernoulli IID traffic. We develop a method based on Lyapunov functions to obtain the delay bounds.

In Chapter 3, we present various implementable high-performance scheduling algorithms. We prove their performance guarantees and discuss implementation details.

Chapter 4 studies a class of switch algorithms under heavy traffic scaling. In order to obtain the state space collapse property of algorithms, we first study the algorithms under fluid scaling. This allows us to obtain two types of results: first, the rate-stability of algorithms; second, the characterization of the state space collapse space. Using the state space collapse space, we obtain a characterization of a delay optimal scheduling algorithm and offer an explanation for the Conjecture 1.

Finally, in Chapter 5 we present the conclusions of the thesis and discuss future research directions.

CHAPTER 2

Maximum Weight Matching

The Maximum Weight Matching(MWM) algorithm has been very well studied in the context of IQ switch scheduling. One of the main reason for the popularity of MWM is the natural association of the switch scheduling problem with bipartite matching problem.

The MWM and its approximation algorithms are central to the study of this thesis. Hence, this chapter is dedicated to the study of properties of MWM and its approximation algorithms. In Section 2.1, we briefly recall the definition and known algorithms to find MWM. We state throughput and delay properties of MWM. We use method based on Lyapunov functions to derive these properties of MWM. The excellent performance of MWM raises the following question: do approximate MWM algorithms have good properties? In Section 2.2, we address this question. We introduce a class of approximate MWM algorithms which we denote by (σ,ρ)-MWM with approximation parameters $\sigma \in \mathcal{Z}_+,\ \rho \in (0,1]$. This notion of (σ,ρ)-MWM is motivated by the theory of approximation algorithms. Again, we use Lyapunov functions based methods to analyze throughput and delay properties of these algorithms. We find the following intuitively pleasing conclusion: a good approximate MWM algorithm also approximates the performance of MWM very well in terms of throughput and delay.

We discuss the strength and weakness of the results of this chapter in Section 2.3. Some of the results that are presented in this section are known. As noted earlier in the Preliminaries,

the background, related work and citations are presented in the bibliographic notes (Section 2.4). We note that, the results of this chapter will be useful throughout this thesis in various contexts.

2.1 The Basic MWM

Consider an $n \times n$ switch operating under the MWM algorithm. Let the arrival process be Bernoulli IID with admissible arrival rate-matrix λ. In this chapter, we only consider Bernoulli IID arrival process. General arrival processes are considered in Chapter 4.

Next, we recall definition of MWM. Let $\pi = [\pi_{ij}] \in \mathbb{P}$ be one of $n!$ possible schedules a scheduling algorithm can choose. Define weight of the schedule π at time m, denoted by $w_\pi(m)$, as

$$w_\pi(m) \quad = \quad Q(m) \cdot \pi = \sum_{ij} \pi_{ij} Q_{ij}(m). \tag{2.1}$$

The MWM algorithm schedules packets according the schedule with the maximum weight. That is, MWM chooses a schedule $\pi^*(m)$ at time m, where

$$\pi^*(m) = \arg\max\{w_\pi(m) : \pi \in \mathbb{P}\}.$$

If there are multiple schedules with the highest weight, then MWM breaks tie arbitrarily. As we shall see later in this thesis, a class of Maximum Weight Matching algorithms is obtained by changing the definition of weight. In this chapter, we focus only on the weight as defined in (2.1). We shall discuss how our results of this chapter change when weight functions differ in section 2.3.

We briefly note that, finding a MWM schedule is well-known algorithmically. There are known polynomial time (in n) algorithms that find MWM (independent of weight). These algorithms are classified as network-flow type algorithms. See Section 2.4 for detailed references.

2.1.1 Properties of MWM

Now, we state the results about throughput and delay property of the MWM.

Theorem 1. *Consider a switch operating under MWM algorithm. Let the arrival process be Bernoulli IID with admissible arrival rate-matrix λ. Then, the switch is strongly stable. Further, the average queue-size is bounded above as*

$$\sum_{ij} E[Q_{ij}] \quad \leq \quad \frac{n^2}{1-\lambda^*}. \tag{2.2}$$

Proof. We first prove the strong stability of the switch under MWM algorithm. To do so, we use the quadratic Lyapunov function, whose value at time m is given as follows.

$$L(Q(m)) \quad = \quad Q(m) \cdot Q(m) = \sum_{ij} Q_{ij}^2(m). \tag{2.3}$$

The results of *Kumar and Meyn* [1995] suggest that to prove strong stability, that is,

$$\limsup_{m\to\infty} E[Q_{ij}(m)] < \infty, \forall i,j,$$

it is sufficient to show that for all time m,

$$E[L(Q(m+1)) - L(Q(m))|Q(m)] \quad \leq \quad -\epsilon\|Q(m)\|_1 + B, \tag{2.4}$$

where ϵ and B are positive constants. We note that, the same conclusion also follows by the Foster's Criteria (see books by *Asmussen* [1987] and *Meyn and Tweedie* [1993a] and works by *Meyn and Tweedie* [1993b] and *Meyn and Tweedie* [1993c] for a detailed exposition on the use of Foster's Criteria).

Now we prove (2.4). Consider the following.

$$\begin{aligned} L(Q(m+1)) - L(Q(m)) \quad &= \quad \sum_{i,j} [Q_{ij}^2(m+1) - Q_{ij}^2(m)] \\ &= \quad \sum_{i,j} [Q_{ij}(m+1) - Q_{ij}(m)][Q_{ij}(m+1) + Q_{ij}(m)]. \end{aligned}$$

Let $\pi^*(m)$ be the schedule used by MWM at time m, $D(m)$ be the induced departures at time m and $A(m+1)$ be the arrivals to queues at time $m+1$.

$$Q_{ij}(m+1) \quad = \quad Q_{ij}(m) + A_{ij}(m+1) - D_{ij}(m), \tag{2.5}$$

$$D_{ij}(m) \quad = \quad \pi_{ij}^*(m)\mathbf{1}_{\{Q_{ij}(m)>0\}}. \tag{2.6}$$

From (2.6), we obtain,

$$\begin{aligned} L(Q(m+1)) - L(Q(m)) &= \sum_{i,j} 2Q_{ij}(m)\left(A_{ij}(m+1) - D_{ij}(m)\right) \\ &+ \sum_{i,j} (A_{ij}(m+1) - D_{ij}(m))^2. \end{aligned} \tag{2.7}$$

Now, in a time slot, at most n packets arrive and n packets depart as well as $(A_{ij}(m) - D_{ij}(m)) \in \{-1, 0, 1\}$. Hence,

$$\sum_{i,j} (A_{ij}(m+1) - D_{ij}(m))^2 \leq 2n. \tag{2.8}$$

Also, (2.6) implies that

$$Q_{ij}(m)D_{ij}(m) = Q_{ij}(m)\pi^*_{ij}(m). \tag{2.9}$$

From (2.7),(2.8) and (2.9), we obtain

$$L(Q(m+1)) - L(Q(m)) \leq \sum_{i,j} 2Q_{ij}(m)\left(A_{ij}(m+1) - \pi^*_{ij}(m)\right) + 2n. \tag{2.10}$$

Taking conditional expectation with respect to $Q(m)$ in (2.10), we obtain

$$\begin{aligned} E\left[L(Q(m+1)) - L(Q(m))|Q(m)\right] &\leq 2\sum_{ij} Q_{ij}(m)\left[E\left[A_{ij}(m+1) - \pi^*_{ij}(m)|Q(m)\right]\right] + 2n \\ &= 2\sum_{ij} Q_{ij}(m)[\lambda_{ij} - \pi^*_{ij}(m)] + 2n \\ &= 2(Q(m)\cdot\lambda - Q(m)\cdot\pi^*(m)) + 2n. \end{aligned} \tag{2.11}$$

We used the fact that arrival process is Bernoulli IID to obtain (2.11).

Now the arrival rate-matrix λ is doubly sub-stochastic. Hence, we can upper bound λ component-wise as

$$\lambda \leq \lambda^* \left(\sum_{k=1}^{n^2} \alpha_k \pi_k\right), \tag{2.12}$$

where for all k, $\pi_k \in \mathbb{P}$, $\alpha_k \in \mathbb{R}_+$ and $\sum_k \alpha_k = 1$.

Also, by property of MWM,

$$Q(m)\cdot\pi \quad \leq \quad Q(m)\cdot\pi^*(m), \quad \forall\pi\in\mathbb{P}. \tag{2.13}$$

From (2.11),(2.12) and (2.13), we obtain

$$E[L(Q(m+1))-L(Q(m))|Q(m)] \quad \leq \quad -2(1-\lambda^*)(Q(m)\cdot\pi^*(m))+2n. \tag{2.14}$$

Now the weight of π^* is at least as large as the average weight of a matching when it is chosen uniformly at random from $\mathbb{P}$. When a matching is chosen uniformly at random, edge (i,j) belong to matching with probability $1/n$. Hence average weight of randomly chosen matching is

$$Q(m)\cdot(1/n)_{ij}=\frac{1}{n}\sum_{i,j}Q_{ij}(m)=\frac{1}{n}\|Q(m)\|_1,$$

where $\|a\|_1=\sum_{ij}a_{ij}$ for $a\in\mathbb{M}_+$. Now, we obtain

$$Q(m)\cdot\pi^*(m) \quad \geq \quad \frac{1}{n}\|Q(m)\|_1. \tag{2.15}$$

From (2.14) and (2.15) we obtain

$$E[L(Q(m+1))-L(Q(m))|Q(m)] \quad \leq \quad -2\frac{(1-\lambda^*)}{n}\|Q(m)\|_1+2n. \tag{2.16}$$

Thus, (2.16) satisfies the desired condition (2.4). This completes the proof of strong stability of MWM algorithm.

Now, we prove the claimed bound on the average queue-size. Consider the following.

$$\begin{aligned} E[L(Q(m+1))] &= E\left\{E[L(Q(m+1))-L(Q(m))|Q(m)]\right\}+E[L(Q(m))] \\ &\leq -2\frac{(1-\lambda^*)}{n}E[\|Q(m)\|_1]+2n+E[L(Q(m))]. \end{aligned} \tag{2.17}$$

Here, the (2.17) follows from (2.16).

Now telescopic summation of (2.17) from $m=0$ to $m=T-1$ and recalling that switch starts empty, we obtain

$$E[L(Q(T))] \quad \leq \quad -2\frac{(1-\lambda^*)}{n}\sum_{m=0}^{T-1}E[\|Q(m)\|_1]+2n. \tag{2.18}$$

Note that, by definition

$$E[L(Q(T))] \quad \geq \quad 0. \tag{2.19}$$

Further, since switch is strongly stable under MWM and $Q(m)$ forms an irreducible, aperiodic Markov chain, it is ergodic and converges to equilibrium distribution. Hence,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{m=0}^{T-1} E[\|Q(m)\|_1] \quad = \quad E[\|Q(\infty)\|_1], \tag{2.20}$$

where $Q(\infty)$ is the queue-size random variable distributed according to its stationary (equilibrium) distribution.

From (2.18),(2.19) and (2.20) we obtain that the stationary average queue-size is bounded above as

$$\sum_{i,j} E[Q_{ij}] \quad \leq \quad \frac{n^2}{1-\lambda^*}. \tag{2.21}$$

This completes the proof of Theorem 1. □

A straightforward corollary of the Theorem 1 is as follows.

Corollary 1. *Consider a switch operating under MWM algorithm. Let the arrival process be Bernoulli with admissible arrival rate-matrix λ. Then, the net stationary average delay is bounded above as*

$$E[D] \quad \leq \quad \frac{n^2}{\lambda_{..}(1-\lambda^*)}. \tag{2.22}$$

Proof. By Little's Law, for any stable system, the average queue-size, $E[Q]$, and average delay, $E[D]$, are related as

$$E[D]\bar{\lambda} \quad = \quad E[Q], \tag{2.23}$$

where $\bar{\lambda}$ is the arrival rate to the system. Now, when the whole switch, when considered as one system, the net queue-size is $\|Q(m)\|_1$ at time m and the net arrival rate is $\lambda_{..} = \sum_{i,j} \lambda_{ij}$. Hence, by Theorem 1 and (2.23), we obtain the statement of Corollary 1. □

2.2 Approximate MWM Algorithms

In this section, we consider a class of approximate MWM algorithms. First, we define these algorithms.

Definition 2 ((σ,ρ)-MWM). *Let $\sigma \in \mathcal{Z}_+$ and $\rho \in (0,1]$. Consider an algorithm $\mathcal{A}$ and let queue-size of switch under this algorithm be $Q(m)$ at time m. Let $\pi^{\mathcal{A}}(m)$ denote the schedule used by algorithm $\mathcal{A}$ at time m. Now, define*

$$\Delta(m) \quad = \quad \max_{\pi \in \mathbb{P}} \{(Q(m) \cdot \pi) - \rho(Q(m) \cdot \pi^{\mathcal{A}})\}. \tag{2.24}$$

Then algorithm $\mathcal{A}$ is called (σ,ρ)-MWM if ($\Delta(m)$,$Q(m)$) is jointly stationary and ergodic as well as $\limsup_{m\to\infty} E[\Delta(m)] \leq \sigma < \infty$.

The above definition of (σ,ρ)-MWM algorithm includes a very wide class of approximation algorithms. Before stating properties of these algorithms, we present few examples of such algorithms that arise naturally.

2.2.1 Examples of (σ,ρ)-MWM

We present two examples of such approximation algorithms. There are many other approximations that naturally arise, either due to simplification of MWM algorithm or due to structure of the problem. In particular, all the algorithms presented in the Chapter 3 belong to this class.

Example 1. *Consider a batch MWM algorithm. Suppose due to the slow logic of a switch, MWM algorithm can compute schedule every K times slots. Thus, algorithm uses queue-size which may be at most K time slots old to compute new schedule. Further, the same schedule is used for K time slots. Now since at most one arrival can occur to each input and at most one arrival can happen to each output, the weight of a matching or schedule change at most by KN in K time slots. Further, the queue-size matrix used to compute a new matching can also be different from actual queue-size matrix by KN packets. Hence, the weight of schedule used by batch MWM is at most $2KN$ less than the weight of MWM. That is, batch MWM is ($2KN$,1)-MWM.*

Example 2. *Consider a well-known greedy maximum weight matching algorithm. The algorithm finds schedule as follows:*

1. *Sort all n^2 queue-sizes in decreasing order.*
2. *Pick the largest queue-size and match corresponding input-output pair.*
3. *Remove all edges incident on this input-output pair.*
4. *Repeat steps 1-3 till no more inputs-outputs are left unmatched.*

It is well known that the weight of greedy maximum weight matching algorithm is at least half the weight of maximum weight matching schedule for the same queue-size. Thus, greedy maximum weight matching is $(0,0.5)$-MWM algorithm.

2.2.2 Properties of (σ,ρ)-MWM

We state the following theorem characterizing the throughput and average queue-size of (σ,ρ)-MWM algorithms.

Theorem 2. *Consider a switch operating under a (σ,ρ)-MWM algorithm. Let the arrival process be Bernoulli with admissible arrival rate-matrix λ. Then, the switch is strongly stable if $\lambda^* < \rho$. Further, when $\lambda^* < \rho$, the stationary average queue-size is bounded above as*

$$\sum_{ij} E[Q_{ij}] \quad \leq \quad \frac{n(n+\sigma)}{\rho - \lambda^*}. \tag{2.25}$$

Proof. The proof is very similar to that of Theorem 1. Let the (σ,ρ)-MWM algorithm under consideration be denoted by $\mathcal{A}$. As in proof of Theorem 1, we first prove the strong stability of algorithm $\mathcal{A}$ and then obtain bound on average queue-size. Now, recall the quadratic Lyapunov function, whose value at time m is

$$L(Q(m)) \quad = \quad Q(m) \cdot Q(m) = \sum_{ij} Q_{ij}^2(m). \tag{2.26}$$

To prove the strong stability under Bernoulli IID arrival process with rate-matrix λ such that $\lambda^* < \rho$, we will show that under these conditions, for all time m,

$$E[L(Q(m+1)) - L(Q(m))|Q(m)] \quad \leq \quad -\epsilon \|Q(m)\|_1 + B, \tag{2.27}$$

where ϵ and B are positive constants.

Following the arguments of the proof of Theorem 1, similar to (2.11), we obtain

$$E\left[L(Q(m+1)) - L(Q(m))|Q(m)\right] \quad \leq \quad 2(Q(m)\cdot\lambda - Q(m)\cdot\pi^{\mathcal{A}}(m)) + 2n, \quad (2.28)$$

where $\pi^{\mathcal{A}}(m)$ is the schedule used by algorithm $\mathcal{A}$ at time m.

By definition of (σ,ρ)-MWM, $\mathcal{A}$ has the property that

$$E[\Delta(m)] \quad \leq \quad \sigma, \quad (2.29)$$

where

$$\Delta(m) \quad = \quad \rho \max_{\pi\in\mathbb{P}} (Q(m)\cdot\pi) - Q(m)\cdot\pi^{\mathcal{A}}(m). \quad (2.30)$$

Now the arrival rate-matrix λ is such that $\lambda^* < \rho$. As before, we can upper bound λ component-wise as

$$\lambda \quad \leq \quad \lambda^* \left(\sum_{k=1}^{n^2} \alpha_k \pi_k\right), \quad (2.31)$$

where for all k, $\pi_k \in \mathbb{P}$, $\alpha_k \in \mathbb{R}_+$ and $\sum_k \alpha_k = 1$.

From (2.28)-(2.31) leads to the following.

$$E\left[L(Q(m+1)) - L(Q(m))|Q(m)\right] \quad \leq \quad -2(\rho-\lambda^*)\max_{\pi\in\mathbb{P}} (Q(m)\cdot\pi) + 2\sigma + 2n. \quad (2.32)$$

We have used stationarity of $\Delta(m)$ in the above inequality.

Using inequality (2.15) along with (2.32), we conclude

$$E[L(Q(m+1)) - L(Q(m))|Q(m)] \quad \leq \quad -2\frac{(\rho-\lambda^*)}{n}\|Q(m)\|_1 + 2(n+\sigma). \quad (2.33)$$

Thus, (2.33) satisfies the desired condition (2.27). This completes the proof of strong stability of (σ,ρ)-MWM algorithm whenever $\lambda^* < \rho$.

Now, we prove the claimed bound on the average queue-size. Similar to arguments of Theorem 1 (i.e. (2.17)-(2.20)) yield the following bound on stationary queue-size.

$$\sum_{i,j} E[Q_{ij}] \quad \leq \quad \frac{n(n+\sigma)}{\rho-\lambda^*}. \quad (2.34)$$

This completes the proof of Theorem 2. □

Similar to Corollary 1, we obtain the following Corollary. We skip the proof as it is exactly the same as Corollary 1.

Corollary 2. *Consider a switch operating under (σ,ρ)-MWM algorithm. Let the arrival process be Bernoulli with admissible arrival rate-matrix λ such that $\lambda^* < \rho$. Then, the net stationary average delay is bounded above as*

$$E[D] \quad \leq \quad \frac{n(n+\sigma)}{\lambda_{..}(\rho - \lambda^*)}. \tag{2.35}$$

2.3 Chapter Summary and Discussion

In this chapter, we studied the throughput and delay property of scheduling algorithms based on MWM algorithm under Bernoulli IID arrival process. We first stated the known stability result about MWM. We obtained bound on the average delay for MWM. Motivated by the definition of approximation algorithms, we define (σ, ρ)-MWM algorithms. We characterized their throughput region and obtained bounds on average delay. The main tool that we utilized to analyze throughput and delay of algorithms is based on the associated Lyapunov function. The method developed in this chapter, especially to bound average delay, is quite general in its scope of application. Though, it requires further work to obtain sharper bounds. Next, we explain the scope of the method and discuss its weakness.

The method is quite general. Given a stable algorithm for which a Lyapunov function is known, the above method gives the bound on average "discrete derivative" of the Lyapunov function as long as the algorithm is MWM with respect to the weight that is equal to the "discrete derivative" of Lyapunov function. This in turn can possibly lead to bounds on average delay. We explain this via the following example.

Example 3. *Consider MWM-2 algorithm where weight of edge (i,j) in switch bipartite graph is $Q_{ij}^2(m)$ at time m. Equivalently, MWM-2 chooses $\pi_2^*(m)$ as schedule at time m, where*

$$\pi_2^* = \arg\max\{\sum_{i,j} Q_{ij}^2 \pi_{ij} : \pi \in \mathbb{P}\}.$$

For this algorithm, consider the following cubic Lyapunov function, whose value at time m is

given as

$$L(Q(m)) \quad = \quad \sum_{i,j} Q_{ij}^3(m). \tag{2.36}$$

Consider the discrete derivative of this cubic Lyapunov function. After substitution and simplification (similar to that in the proofs of Theorems 1 and 2), we obtain

$$L(Q(m+1)) - L(Q(m)) \quad = \quad \sum_{i,j} 3Q_{ij}^2(m)\delta_{ij}(m) + 3Q_{ij}(m)\delta_{ij}^2(m) + \delta_{ij}^3(m), \tag{2.37}$$

where $\delta_{ij}(m) = A_{ij}(m+1) - D_{ij}(m)$. *Considering two cases: (i)* $Q^* = \max_{i,j} Q_{ij} > \frac{2\phi n}{\epsilon}$ *and (ii)* $Q^* = \max_{ij} Q_{ij} > \frac{2\phi n}{\epsilon}$, *for any* $\phi > 1$ *and* $\epsilon = (1-\lambda^*)$. *Now since MWM-2 chooses schedule that maximizes the weight, where weight is quadratic queue-size, we obtain the following crude bound on (2.37).*

$$\begin{aligned} L(Q(m+1)) - L(Q(m)) \quad &\leq \quad \mathbf{1}_{\{Q^* > \frac{2\phi n^2}{\epsilon}\}} \left(-\frac{3\epsilon(1-1/\phi)}{n} \sum_{i,j} Q_{ij}^2(m) \right) + 12\phi n^3/\epsilon + 2n \\ &\leq \quad \left(-\frac{3\epsilon(1-1/\phi)}{n} \sum_{i,j} Q_{ij}^2(m) \right) + 12\phi n^3/\epsilon + 2n. \end{aligned} \tag{2.38}$$

This in turn will lead to the following bound on stationary queue-size.

$$\sum_{i,j} E[Q_{ij}^2] \quad \leq \quad \frac{4\phi n^3 + 2n/3}{(1-\lambda^*)(1-1/\phi)}. \tag{2.39}$$

Certainly, (2.39) can imply bound on $\sum_{i,j} E[Q_{ij}]$. *Moreover, the bound (2.39) itself is interesting.*

The weakness of this method is the same as the strength: its too general. Due to its generality, it provides weaker bounds. To expose the weakness, we obtain a direct bound on a trivial Random algorithm. This bound turns out to be better than that of Theorem 1 ! Certainly, we believe the MWM is better than Random algorithm.

Example 4. *The Random algorithm does the following: every time, pick a matching* π *uniformly at random from all* $n!$ *matching of* $\mathbb{P}$ *and use it as the schedule. Under Bernoulli IID uniform traffic, the arrival rates are such that,* $\lambda_{ij} = \frac{\lambda^*}{n}, \forall i, j$. *Under Random algorithm, the*

probability that queue (i,j), is serviced at any time is $1/n$ independent of every time. Thus, each queue $Q_{ij}(m)$ has Bernoulli IID arrival process of rate λ^*/n and Bernoulli IID service process of rate $1/n$. The average queue-size of such a queue is a well-known queuing fact, which is as follows.

$$E[Q] = \frac{\lambda^*(n-1)}{n(1-\lambda^*)}. \tag{2.40}$$

Summing over all n^2 queues, we obtain,

$$\sum_{i,j} E[Q_{ij}] = \frac{n(n-1)\lambda^*}{1-\lambda^*}. \tag{2.41}$$

A straightforward comparison of bounds from Theorem 1 and (2.41) shows that (2.41) is smaller. We strongly believe that Random is not better than MWM. Thus, exposes weakness of our method.

The main outcome of this chapter is the following: MWM and its approximations have very good throughput and delay property. This makes them very attractive scheduling algorithms for the purpose of implementation. Unfortunately, due to the implementation concerns, MWM or its known approximations are not not feasible to implement. In the next chapter, we will present new design techniques to obtain simple-to-implement approximations of MWM.

2.4 Bibliographic Notes

The problem of finding MWM can be posed as a Linear program. Hence, for example, Simplex Algorithm can be used to find MWM *Dantzig* [1963]. However, it may not find MWM scheduling in polynomial time (in n). In 1970s and 1980s, a lot of interesting work was done in the field of Combinatorial algorithms to find good algorithm for MWM. Notably, an algorithm based on the results of Edmonds and Karp finds MWM in $O(n^3)$ time (see works by *Edmonds* [1965] and *Edmonds and Karp* [1972]). This algorithm along with many other related network flow algorithms can be found in monograph by *Tarjan* [1983].

The result about stability of MWM under Bernoulli IID traffic was first established by *McKeown et al.* [1996]. The results of *Tassiulas and Ephremides* [1992] in the context of Radio hop networks imply these results. Both of these results used quadratic Lyapunov function in order to achieve throughput results.

The definition of (σ,ρ)-MWM is motivated from the classical notion of competitive ratio for online algorithm which was first introduced by *Sleator and Tarjan* [1985]. Our main contributions in this Chapter are the method for obtaining bound on average delay and the study of (σ,ρ)-MWM. These results are primarily based on work by *Shah and Kopikare* [2002]. Initial results on obtaining delay bounds for MWM was done by *Leonardi et al.* [2001] using somewhat different method. Though, their results are qualitatively very similar, their method does not extend as well as our method. Another application of the method of this chapter can be found in work by *Shah* [2003].

Historically, obtaining bounds on delay or queue-size for complex queuing system has been central to the study of stochastic networks. There are known results in past that utilized Lyapunov function to obtain bounds on delay. For example, see works by *Hajek* [1982], *Kumar and Kumar* [1994].

CHAPTER 3

Implementable High-Performance Algorithms

This chapter presents simple-to-implement and high-performance scheduling algorithms for IQ switches. The results of Chapter 2 show that Maximum Weight Matching has maximal throughput and low packet delay. This makes MWM a very attractive algorithm. However, MWM is not implementable for the following reasons: the best known algorithm to find Maximum Weight Matching requires $O(n^3)$ operations in the worst case. For example, for a 30 port switch, it will require 27000 operations. Thus, a switch operating at 10Gbps, with packet size of 50 bytes will be required to do so many operations roughly every 5-10ns. This is infeasible under current technology. Further, due back-tracking type routine involved in such algorithm, it is not suitable for pipelining. Similar to MWM, other known good algorithms are very difficult to implement. This leads us to the following questions: is it possible for an algorithm to compete with the throughput and delay performance of MWM and yet be simple to implement? if yes, what feature of the scheduling problem should be exploited?

In this chapter, we answer the above questions in affirmative by exploiting the following features: (1) *Randomization*: in a variety of situations where the scalability of deterministic algorithms is poor, randomized algorithms are easier to implement and provide a surprisingly good performance. (2) *Using memory*: the state of the switch, that is queue-lengths, change very little during successive time slots. Hence, a heavy matching will continue to be heavy over

a few time slots, suggesting that carrying some information, or retaining memory, between iterations should help simplify the implementation while maintaining a high level of performance. (3) *Using arrivals*: since the increase in queue-lengths is entirely due to arrivals, knowledge of recent arrivals can be useful in finding a heavy matching. (4) *Hardware parallelism*: finding heavy matchings essentially involves a search procedure, requiring a comparison of the weight of several matchings. The natural structure on the space of matchings allows use of parallelism in hardware to conduct this search efficiently.

The rest of the chapter presents algorithms exploiting the above observations and novel methods to analyze their performance. The Section 3.1 discusses use of randomization and memory to obtain a very simple stable algorithm. We show that randomization alone is not useful to obtain stable algorithm. But, combining randomization with memory yields a stable algorithm. A derandomization of this algorithm is also stable. Though, these simple algorithms are stable, they have very poor average delay compared to MWM. To improve delay, we present algorithms LAURA, SERENA and APSARA in Section 3.2. We study throughput and delay properties of these algorithms theoretically and via extensive simulation study. Our results show that all of these algorithms perform very competitively with respect to MWM. In Section 3.3, we discuss implementation details of these algorithms. Finally, we present bibliographic notes related to this chapter in Section 3.4.

3.1 Stable Randomized Algorithms

Randomized algorithms are particularly simple to implement because they work on a few randomly chosen samples rather than on the whole state space. The MWM finds, from amongst the $n!$ possible permutations of $\mathbb{P}$, that permutation whose weight is the highest. An obvious randomization of MWM yields the following algorithm, which we denote by Algo1: *At each time m, let the permutation used by* Algo1 *be the heaviest of d $(d > 1)$ permutations chosen uniformly at random from $\mathbb{P}$.*

For simplicity, we want to have small d. Unfortunately, the following theorem shows that Algo1 is not stable, even when $d = \Theta(n)$.

Theorem 3. *For any $d \leq cn$, where $c > 0$,* Algo1 *is not stable.*

Proof. Consider the queue at input i for output j. This queue is served, that is, input i is matched to output j at time m, only if input i is matched to output j by at least one of the

d randomly chosen permutations or matchings. Consider the following.

$$\begin{aligned} p_{ij} &= \Pr(i \text{ is matched to } j \text{ in one of the } d \text{ random s}) \\ &= 1 - \Pr(i \text{ is not matched to } j \text{ in any of the } d \text{ random matchings}) \\ &= 1 - \Pr(i \text{ is not matched to } j \text{ in one random matching})^d \\ &= 1 - \left(1 - \frac{1}{n}\right)^d \\ &\leq 1 - \left(1 - \frac{1}{n}\right)^{cn} \quad \text{for } d \leq cn \\ &\to 1 - e^{-c}. \end{aligned}$$

Therefore, the service rate available for packets from input i to output j is at most $1-e^{-c} < 1$. And, as soon as $\lambda_{ij} > 1 - e^{-c}$, we have that the switch is unstable under Algo1. □

Remark: Note that the above theorem has a much stronger implication: *Any* scheduling algorithm that only uses $d = O(n)$ random matchings cannot achieve 100% throughput. Further, there is no assumption about the distribution of the packet arrival process, only a rate assumption. This adds strength to the next algorithm, Algo2, due to *Tassiulas* [1998].

3.1.1 Algo2: Randomized Algorithm with Memory

The Algo2 uses randomization with memory. It is described as follows.

Algo2:

(a) Let $\pi(m)$ be the schedule used at time m.

(b) At time $m+1$ choose a matching $\pi^r(m+1)$ uniformly at random from the set of all $n!$ possible matchings.

(c) Let $\pi(m+1) = \arg \max_{\pi \in \{\pi(m), \pi^r(m+1)\}} \pi \cdot Q(m+1)$.

The following theorem states that Algo2 is stable.

Theorem 4. Algo2 *is stable under any Bernoulli IID arrival process with admissible arrival rate-matrix* λ *and the average queue-size is bounded above as*

$$\sum_{ij} E[Q_{ij}] \leq \frac{n(n+n!)}{1-\lambda^*}. \tag{3.1}$$

Proof. By definition, the number of packets arriving in a time slot is at most n and the number of packets departing in a time slot is at most n. Hence, weight of a permutation can change by at most $2n$ in a time slot.

Under algorithm Algo2, at time m, let $\pi(m)$ denote the schedule used and $Q(m)$ be the queue-size matrix. Let the corresponding MWM schedule be $\pi^*(m)$ at time m, that is,

$$\pi^*(m) = \arg\max_{\pi \in \mathbb{P}} \pi \cdot Q(m).$$

From the above observation, for any $\ell \leq m$, it is easy to see that

$$|\pi^*(\ell) \cdot Q(\ell) - \pi^*(m) \cdot Q(m)| \quad \leq \quad 2n(m-\ell). \tag{3.2}$$

Due to the use of memory in Algo2, it is easy to see that

$$\pi(m) \cdot Q(m) \quad \geq \quad \pi(m-1) \cdot Q(m-1) - 2n. \tag{3.3}$$

Let,

$$M = \inf\{\ell \geq 0 : \pi(m-\ell) = \pi^*(m-\ell)\}.$$

Combining (3.2) and (3.3), we obtain

$$\pi(m) \cdot Q(m) \quad \geq \quad \pi^*(m) \cdot Q(m) - 4Mn. \tag{3.4}$$

Due to independent drawing of random permutation every time under Algo2, M is upper bounded by a Geometric random variable with probability $1/n!$. Hence,

$$E[M] \quad \leq \quad n!. \tag{3.5}$$

From (3.4) and (3.5), we obtain that Algo2 is ($n!$,1)-MWM. Hence, the statement of Theorem 4 follows from the Theorem 2 of Chapter 2. □

3.1.2 Algo3: Derandomization of Algo2

The algorithm Algo2 uses external randomization. Next, we consider a derandomization of this algorithm, which we call Algo3. Before presenting the algorithm we need the concept of a Hamiltonian walk on a $\mathbb{P}$. Consider a complete graph with $n!$ nodes, each corresponding

to a distinct $\pi \in \mathbb{P}$. Let $H(k)$ denote a Hamiltonian walk on this graph; that is, $H(k)$ visits each of the $n!$ distinct nodes exactly once during times $k = 0, \ldots, n!–1$. We extend $H(k)$ for $k \geq n!$ by defining $H(k) = H(k \mod n!)$. One simple algorithm for such a Hamiltonian walk is described, for example, in Chapter 7 of *Nijenhuis and Wilf* [1978]. This is a very simple algorithm that requires $O(1)$ space and $O(1)$ time, to generate $H(k+1)$ given $H(k)$. Under this algorithm $H(k)$ and $H(k+1)$ differ in exactly two edges. Consider the following example of this algorithm.

Example 5. *Let $n = 3$. The algorithm generates the following Hamiltonian walk on $\mathbb{P}$: $H(0) = (1,2,3)^*$, $H(1) = (1,3,2)$, $H(2) = (3,1,2)$, $H(3) = (3,2,1)$, $H(4) = (2,3,1)$, $H(5) = (2,1,3)$, $H(6) = H(0)$, and $H(7) = H(1)$, and so on.*

Now, we describe the algorithm.

Algo3:

(a) Let $\pi(m)$ be the schedule used at time m.

(b) Let $H(m) \in \mathbb{P}$ be permutation corresponding to the Hamiltonian walk on the graph corresponding to $\mathbb{P}$.

(b) Let $\pi(m+1) = \arg \max_{\pi \in \{\pi(m), H(m+1)\}} \pi \cdot Q(m+1)$.

Next, we state the properties of Algo3, very similar to that of Algo2.

Theorem 5. Algo3 *is stable under any Bernoulli IID arrival process with admissible arrival rate-matrix λ and the average queue-size is bounded above as*

$$\sum_{ij} E[Q_{ij}] \quad \leq \quad \frac{n(n+n!)}{1-\lambda^*}. \tag{3.6}$$

Proof. The proof is very similar to that of Theorem 4. Under algorithm Algo3, at time m, let $\pi(m)$ denote the schedule used and $Q(m)$ be the queue-size matrix. Let the correspond MWM schedule be $\pi^*(m)$ at time m, that is,

$$\pi^*(m) = \arg \max_{\pi \in \mathbb{P}} \pi \cdot Q(m).$$

*Here, by $\pi = (\pi(1), \pi(2), \pi(3))$, we mean that i is matched to $\pi(i)$, for $i = 1, 2, 3$, under permutation π.

As observed in the proof of Theorem 4, the weight of a schedule changes by at most $2n$ in successive time slots. Hence,

$$|\pi^*(m) \cdot Q(\ell) - \pi^*(m) \cdot Q(m)| \quad \leq \quad 2n(m-\ell). \tag{3.7}$$

Let,

$$M = \inf\{\ell \geq 0 : H(m-\ell) = \pi^*(m)\}.$$

By property of Algo3,

$$\begin{aligned} \pi(m-M) \cdot Q(m-M) &\geq H(m-M) \cdot Q(m-M) \\ &= \pi^*(m) \cdot Q(m-M) \\ &\geq \pi^*(m) \cdot Q(m) - 2nM, \end{aligned} \tag{3.8}$$

where the last inequality follows from (3.7).

Due to the use of memory in Algo3, it is easy to see that

$$\pi(m) \cdot Q(m) \quad \geq \quad \pi(m-M) \cdot Q(m-M) - 2nM. \tag{3.9}$$

Combining (3.8) and (3.9), we obtain

$$\pi(m) \cdot Q(m) \quad \geq \quad \pi^*(m) \cdot Q(m) - 4Mn. \tag{3.10}$$

Since $H(\cdot)$ covers all permutations in $n!$ time, $M \leq n!$. Hence, from (3.10) we obtain that Algo3 is ($n!$,1)-MWM. Hence, the statement of Theorem 4 follows from the Theorem 2 of Chapter 2. □

3.1.3 Delay of Algorithms Algo2 and Algo3

The above algorithms, Algo2 and Algo3 are extremely simple and Theorems 4 and 5 prove their stability. But the delay induced by these algorithms are too large. We present an example simulation study to exhibit this claim.

Simulation Setup

We describe the simulation setup, which is used for all the simulation results presented in the rest of the chapter.

Switch: Number of ports, $n = 32$. Each VOQ can store up to 10,000 packets. Excess packets are dropped.

Input Traffic: All inputs are equally loaded on a normalized scale, and $\rho \in (0, 1)$ denotes the normalized load. The arrival process is Bernoulli IID. Let $|k| = (k \mod n)$. The following load matrices are used to test the performance of algorithms.

1. **Uniform:** The arrival rate-matrix λ is such that, $\lambda_{ij} = \rho/n \ \ \forall i, j$. This is a very friendly type of traffic.

2. **Diagonal:** The arrival rate-matrix λ is such that: $\lambda_{ii} = 2\rho/3$, $\lambda_{i|i+1|} = \rho/3 \ \ \forall i$ and $\lambda_{ij} = 0$ for all other i, j. This is a very skewed loading, in the sense that input i has packets only for outputs i and $|i + 1|$. This traffic loading tests algorithms very well.

Performance measures: We compare the average queue-lengths induced by different algorithms. The simulations are run until the estimate of the average delay reaches the relative width of the confidence interval equal to 1% with probability $\geq 95\%$. The estimation of the confidence interval width uses the *batch means* approach.

Results

Figure 3.1 plots the average queue-size induced under Algo2 and MWM under Diagonal traffic pattern. The Y-axis is the average queue-size (logarithmic scale) and the X-axis is the load ρ. The figure shows that MWM has very low average queue-size even when ρ is near 1. On the contrary, Algo2 has very large average queue-size even at $\rho = 0.4$ and it becomes too large beyond $\rho = 0.4$ and hence not plotted in the figure. The figure also plots performance of known heuristics iSLIP and iLQF for comparison. Note that though these heuristics are known to be unstable, they perform much better than Algo2, exposing its poor performance. For completeness, we note that all algorithms perform equally well under Uniform traffic.

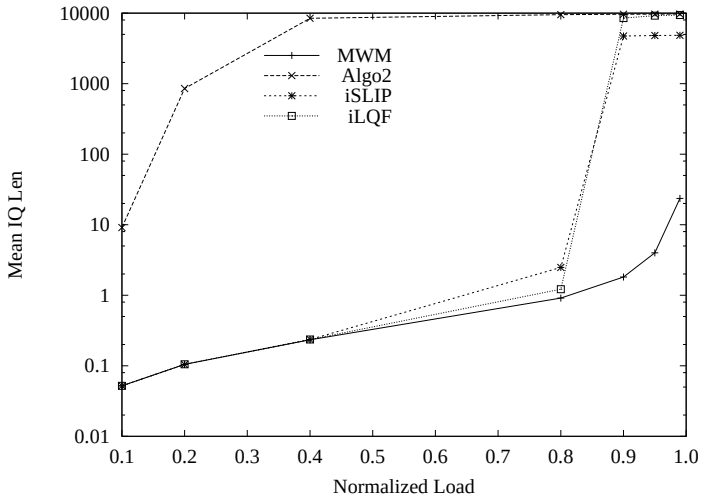

Figure 3.1: Performance of Algo2 under Diagonal traffic.

3.2 Low Delay Algorithms

The Algo2 and Algo3 suggest that achieving 100% throughput is not difficult. On the contrary, to reduce delay, an algorithm has to do extra work. In this section, we describe three different algorithms that respectively use parallelism, randomization and the information in arrivals to achieve 100% throughput and low delay.

3.2.1 APSARA: Use of Parallelism

As noted in the introduction, determining the maximum weight matching essentially involves a search procedure, which can take many iterations and be time-consuming. Since our goal is to design high-performance schedulers for high speed switches, algorithms that involve too many iterations are unattractive.

We wish to design a high-performance scheduler that only requires a *single* iteration. Therefore, we must devise a fast method for finding good schedules. One method for speeding up the scheduling process is to search the space matchings in parallel. Fortunately, the space of matchings has a nice combinatorial structure which can be exploited for conducting efficient searches. In particular, it is possible to query the "neighbors" of the current matching in parallel and use the heaviest of these as the matching for the next time slot. This observation

inspires the APSARA algorithm, which employs two ideas: (1) Use of memory, and (2) exploring neighbors in parallel, where neighbors are defined such that it is easy to compute them using hardware parallelism.

Definition 3 (Neighbor). *Given a permutation π, a permutation π' is said to be a neighbor of π iff there exists $i_1, i_2 \in \{1, \ldots, n\}$, such that the following is satisfied: (1) $\pi(i_1) = \pi'(i_2)$, (2) $\pi(i_2) = \pi'(i_1)$ and (3) $\pi(i) = \pi'(i)$, for $i \neq i_1, i_2$. The set of all neighbors of π is denoted $\mathcal{N}(\pi)$, whose cardinality is $\binom{n}{2}$.*

APSARA: The Basic Algorithm

Let $\pi(m)$ be the matching determined by APSARA at time m. Let $H(m+1)$ the matching corresponding to the Hamiltonian walk at time $m+1$. At time $m+1$ APSARA does the following:

(i) Determine $\mathcal{N}(\pi(m))$ and $H(m+1)$.

(ii) Let $\mathcal{M}(t+1) = \mathcal{N}(\pi(m)) \cup \{H(m+1)\} \cup \{\pi(m)\}$.

(iii) The schedule at time $m+1$ is given by:

$$\pi(m+1) = \arg \max_{\pi' \in \mathcal{M}(m+1)} \pi' \cdot Q(m+1).$$

APSARA requires the computation of the weight of neighbor matchings. Each such computation is easy to implement since a neighbor π' differs from the matching $\pi(m)$ in exactly two edges. However, computing the weights of all $\binom{n}{2}$ neighbors requires a lot of space in hardware for large values of n.

APSARA-L: Deterministic Approximation

To reduce the number of neighbors from $\binom{n}{2}$ to n, we consider the following neighbor-set.

Definition 4 (Linear-Neighbor). *Given a permutation π, a permutation π' is said to be a linear-neighbor of π iff there there exists $i \in \{1, \ldots, n\}$ such that the following is satisfied: (1) $\pi(i) = \pi'((i \mod n) + 1)$, (2) $\pi((i \mod n) + 1) = \pi'(i)$ and (3) $\pi(j) = \pi'(j)$, for $j \neq i$. The set of all neighbors of π is denoted $\mathcal{N}_L(\pi)$, whose cardinality is n.*

Denote by APSARA-L the version of the basic APSARA algorithm when neighbors are chosen from $\mathcal{N}_L(\pi)$.

APSARA-R(K): Randomized Approximation

Suppose hardware constraints only allow us to query K neighbors. Let $\mathcal{N}_K(\pi)$ denote the set of K permutations picked uniformly at random from the set $\mathcal{N}(\pi)$. The randomized version of APSARA algorithm, denoted by APSARA-R, works with $\mathcal{N}_K(\pi)$ to determine its schedule. Note that, if $K = \binom{n}{2}$ then APSARA-R defaults to APSARA.

Remark: Note that, APSARA (and its variants) generate all the matchings in the neighborhood set oblivious to the current queue-lengths. The queue-lengths are only used to select the heaviest matching from the neighborhood set. It is therefore possible that the matching determined by APSARA, while being heavy, is not of maximal size. That is, there exists an input, say i, which has packets for an output j, but the matching chosen by algorithm, both i and j are connected via empty (0 weight) edges. To overcome this unnecessary idleness, one possible way is the following: complete the matching determined by APSARA in a round-robin order over the input-output ports that are empty. This version of the algorithm is called MaxAPSARA.

Properties of APSARA

The APSARA algorithm and its variants are stable as stated below.

Theorem 6. *The algorithms* APSARA, APSARA-L *and* APSARA-R *are all stable under Bernoulli IID arrival process with admissible arrival rate-matrix λ. Further, the average queue-size is bounded as*

$$\sum_{i,j} E[Q_{ij}] \quad \leq \quad \frac{n(n+n!)}{1-\lambda^*}. \tag{3.11}$$

Proof. All versions use the Hamiltonian walk, $H(\cdot)$ and are based on using memory. Therefore, the proof of Theorem 5 implies the statement of Theorem 6. □

Theorem 6 does not suggest why APSARA and its variants should do much better than Algo2 or Algo3. The following property indicates why APSARA should be better.

Theorem 7. *Let $\pi(m)$ denote the schedule obtained by* APSARA *at time m. If $\pi(m) = \pi(m-1)$, that is the schedule does not change from time $m-1$ to time m, then*

$$\pi(m) \cdot Q(m) \quad \geq \quad \frac{1}{2} \max_{\pi \in \mathbb{P}} \pi \cdot Q(m). \tag{3.12}$$

Proof. Without loss of generality, let the identity permutation be the one that maximizes $\pi \cdot Q(m)$ for all $\pi \in \mathbb{P}$, that is, identity permutation is the maximum weight matching at time m. As noted before, $\pi(m)$ is the permutation chosen by APSARA and $\pi(m) = \pi(m-1)$.

Now, consider any $i \in \{1, \ldots, n\}$. By definition, $\pi(m)$ connects i to $\pi(m)(i)$. Let $\mathcal{I}_1 = \{i : \ i = \pi(m)(i)\}$. Let $\mathcal{I}_2 = \{1, \ldots, n\} - \mathcal{I}_1$. Now, for all $i \in \mathcal{I}_1$,

$$Q_{i\pi(m)(i)}(m) \quad = \quad Q_{ii}(m). \tag{3.13}$$

Now consider $i \in \mathcal{I}_2$. Since $\pi(m) = \pi(m-1)$, by the property of APSARA, it follows that

$$Q_{i\ \pi(m)(i)}(m) + Q_{\pi^{-1}(m)(i)\ i}(m) \geq Q_{ii}(m). \tag{3.14}$$

Now summing over i, from (3.13) and (3.14), it is easy to deduce that

$$\sum_i Q_{i\pi(m)(i)}(m) + Q_{\pi^{-1}(m)(i)i}(m) \quad \geq \quad \sum_i Q_{ii}(m). \tag{3.15}$$

Since $\pi(m)$ is a permutation, $\sum_i Q_{i\pi(m)(i)}(m) = \sum_i Q_{\pi^{-1}(m)(i)i}(m)$. Further, by definition

$$\pi(m) \cdot Q(m) \quad = \quad \sum_i Q_{i\pi(m)(i)}(m). \tag{3.16}$$

From (3.15), (3.16) and recalling that identity permutation corresponds to the maximum weight matching at time m, the statement of the Theorem 7 follows. □

Simulation Results

We study performance of APSARA and its variants via extensive simulation study. The simulation setup is identical to the one described in Section 3.1.3 of this chapter.

Figure 3.2 compares the average queue-sizes induced by APSARA, MWM, iSLIP (with n iterations) and iLQF (with n iterations) under Diagonal traffic. The figure suggests that

APSARA and MaxAPSARA perform very competitively with MWM under all loadings. On the other hand, both iLQF and iSLIP incur severe packet losses and large delays under heavy loading. We also note that under low loads, APSARA deviates from MaxAPSARA. The reason is as follows: since APSARA is not maximal, it may cause few queues to idle and at small loads maximality is really what is important (as showed by iSLIP and iLQF's performance). Note that, the difference is negligible – its no more than 10 packets on average. The figure also shows that though APSARA-L has only 32 neighbors, it performs quite well compared to APSARA, which uses $\binom{32}{2} = 496$ neighbors. Separately, the randomized variant APSARA-R(32) does not perform as well as APSARA-L. Other study leads us to recommend the following: when the number of allowable neighbors $K \ll n$, only then use randomized version of APSARA-R(32).

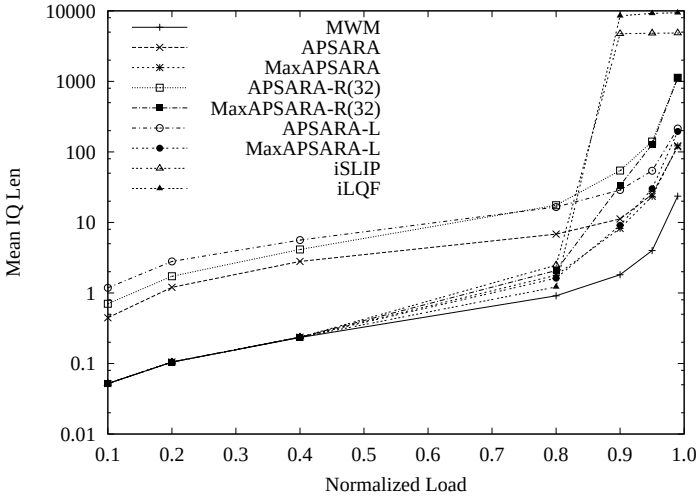

Figure 3.2: Performance of APSARA under Diagonal traffic.

3.2.2 LAURA: Use of Problem Structure

As shown in Section 3.1, the Algo2 provides 100% throughput. However, its delay performance is quite poor. This is mainly because of the following reason: every time, Algo2 selects one of the two matchings (one random and the other from previous time) in its entirety rather than selecting heavy edges from both while satisfying matching constraints. The algorithm LAURA is mainly based on this observation. It uses a procedure called Merge to obtain a

heavier matching than given two matching by selecting heavier edges from both matchings. As we shall see, the Merge procedure cleverly uses the structure of the problem, leading to a very simple implementation. In addition, a non-uniform random sampling is used to bias a random sample towards heavier matchings. Thus, the main features used in the design of LAURA are: (1) use of memory, (2) non-uniform random sampling, and (3) a Merge procedure to obtain a better matching.

LAURA Algorithm

Let $\pi(m)$ be the matching used by LAURA at time m. At time $m+1$ LAURA does the following:

(a) Generate a random matching $\pi^r(m+1)$ using the Random procedure.

(b) Use $\pi(m+1) = \text{Merge}(\pi^r(m+1), \pi(m))$ as the schedule for time $m+1$.

Now we describe the Random and Merge procedures.

The Random Procedure

Let $\mathcal{F}_\eta(\pi)$ denote the minimal set of edges in the matching π carrying at least a fraction η $(0 \leq \eta \leq 1)$ of its weight. We shall call η the *selection factor*.

Random is the following iterative procedure: Initially, all inputs and outputs are marked as *unmatched*. The following steps are repeated in each of I iterations, where I is typically $\log_2 n$:

(i) Let i be the current iteration number. Let $k \leq n$ be the number of *unmatched* input-output pairs. Out of the $k!$ possible matchings between these unmatched input-output pairs, a matching $\pi_i(k)$ is chosen uniformly at random.

(ii) If $i < I$, retain the edges corresponding to $\mathcal{F}_\eta(\pi_i(k))$ and mark the nodes they cover as *matched*. If $i = I$, then retain all edges of $\pi_i(k)$.

The Merge Procedure

Consider a switch bipartite graph with Q matrix as its edge weights. Given two matchings $\pi(1)$ and $\pi(2)$, define

$$\mathcal{S}(\pi(1), \pi(2)) = \{\pi \in \mathbb{P} : \pi_{ij} = 1 \text{ only if } \pi(1)_{ij} = 1 \text{ or } \pi(2)_{ij} = 1\}.$$

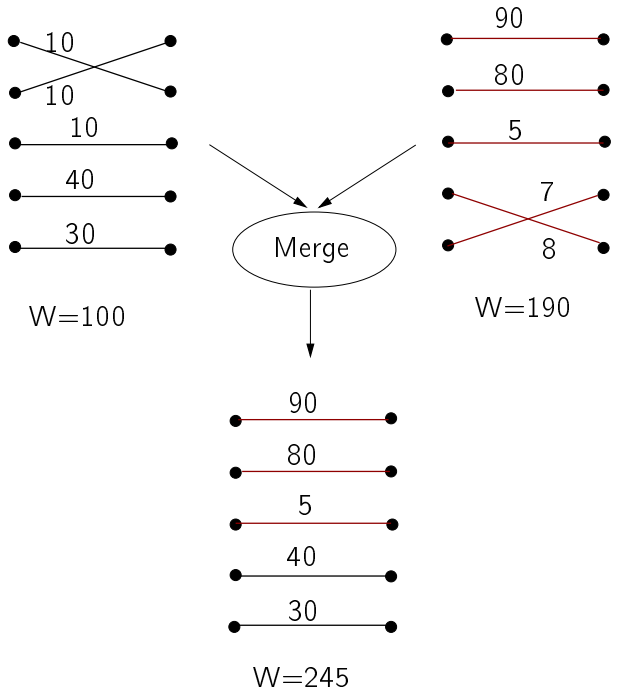

Figure 3.3: An example of Merge procedure.

The Merge procedure, when applied to $\pi(1)$ and $\pi(2)$, it returns a matching $\tilde{\pi}$ such that

$$\tilde{\pi} \quad = \quad \arg \max_{\pi \in \mathcal{S}(\pi(1),\pi(2))} \{\pi \cdot Q\}. \tag{3.17}$$

The Merge finds such matching using only $2n$ addition and subtraction. It is described as follows: Color the edges of $\pi(1)$ as red and the edges of $\pi(2)$ as green. Start at output node j_1 and follow the red edge to an input node, say i_1. From input node i_1 follow the (only) green edge to its output node, say j_2. If $j_2 = j_1$, stop. Else continue to trace a path of alternating red and green edges until j_1 is visited again. This gives a "cycle" in the subgraph of red and green edges.

Suppose the above cycle does not cover all the red and green edges. Then there exists an output j outside this cycle. Starting from j repeat the above procedure to find another cycle. In this fashion find all cycles of red and green edges. Suppose there are ℓ cycles, $C_1, ..., C_\ell$ at the end. Then each cycle, C_i, contains two matchings: G_i which has only green edges, and R_i which has only red edges. For each cycle C_i, the Merge chooses R_i if the sum of the queue-size corresponding to these edges is higher than that of the G_i. Else, Merge chooses

G_i. It is easy to show that the final matching as chosen above is precisely the one claimed in (3.17). Figure 3.3 illustrates the Merge procedure.

Properties of LAURA

The following theorem is about the throughput and delay property of LAURA.

Theorem 8. *The algorithm* LAURA *is stable under Bernoulli IID arrival process with admissible arrival rate-matrix λ. Further, the average queue-size is bounded as*

$$\sum_{i,j} E[Q_{ij}] \quad \leq \quad \frac{n(n+n!)}{1-\lambda^*}. \tag{3.18}$$

Proof. Consider the following two facts about LAURA: (1) The probability of LAURA choosing maximum weight matching schedule is at least $1/n!$, every time, independent of everything else and (2) LAURA uses memory.

Now, in the proof of Theorem 4, we showed that Algo2 is ($n!$,1)-MWM using the above two properties. Hence, Theorem 2 of chapter 2 implies the statement of Theorem 8. □

Simulation Results

We study the performance of LAURA via extensive simulation study. The simulation setup is identical to the one described in Section 3.1.3 of this chapter.

We set the selection factor $\eta = 0.5$, and the number of iterations $I = 5 = \log_2 n$, for $n = 32$. The average queue-size induced under algorithm LAURA is compared with that of the MWM, iSLIP, iLQF and Algo2 algorithms under Diagonal traffic. The results are shown in Figure 3.4. The algorithms LAURA and MaxLAURA (which is maximal version of LAURA) perform quite competitively with respect to MWM. We see that iSLIP and iLQF suffer large packet losses at high loads. Strangely enough, although Algo2 is provably stable (as opposed to iSLIP and iLQF), its performance in terms of average backlog is the worst.

The Impact of Merge

In this section we study the role of the Merge procedure in LAURA for obtaining good delay performance. For this purpose, we consider the following two algorithms: Algo2 as described in Section 3.1 and its variant using Merge, denoted by Algo4. At time $m+1$, the Algo4 uses

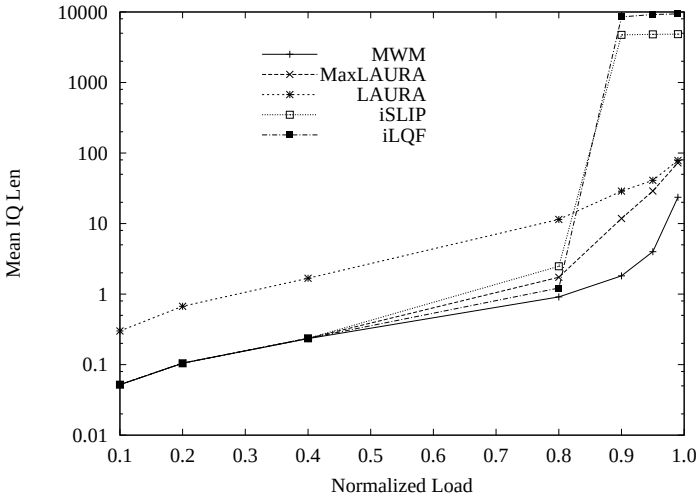

Figure 3.4: Performance of LAURA under Diagonal traffic.

schedule $\pi(m+1) = \text{Merge}(\pi^r(m+1), \pi(m))$, where $\pi(m)$ is schedule used at time m and $\pi^r(m+1)$ is a matching chosen uniformly at random at time $m+1$.

Figure 3.5 show the average queue lengths for these two algorithms under Diagonal traffic. We note that both algorithms behave almost the same under Uniform traffic, and thus the Merge procedure does not make a big difference to the performance under this traffic. When the traffic is not Uniform, as shown in Figure 3.5, Algo4 performs much better compared to Algo2. This shows that the use of the Merge procedure is essential for obtaining good delay performance under non-uniform traffic.

Learning Time: Merge v/s Max

The main reason behind achieving 100% throughput for algorithms like Algo2 and Algo4 is the finite amount of time (on average) that it takes for these algorithms to obtain a matching whose weight is comparable to that of MWM. But the learning time can be drastically different and it directly affects the delay performance of the underlying scheduling algorithm.

We now make a comparison of the learning time of Algo4, which uses Merge procedure, with that Algo2, which uses Max procedure. First we present the simulation study under different scenarios and then present analytical results to understand the observed behavior

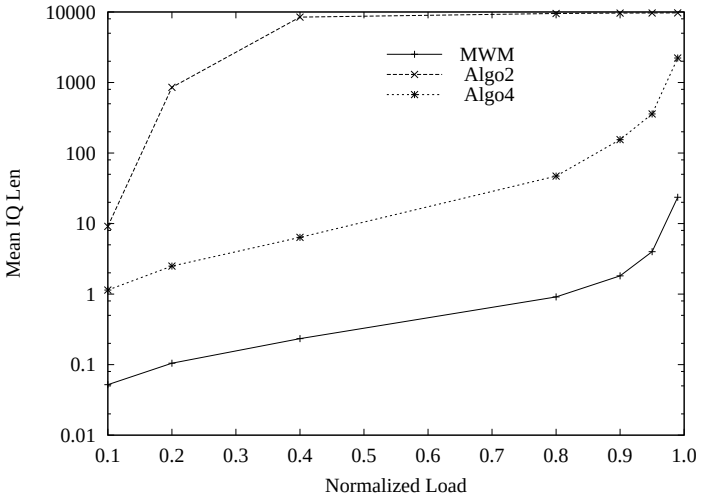

Figure 3.5: An illustration of impact of Merge on Performance.

under a simple model.

Comparison Algo2 *and* Algo4*: Simulation*

Simulation setting: A random weighted bipartite graph is created by choosing the weight of each edge according to independent and identically distributed random variables with mean 1. We consider three different distributions : (a) Exponential, (b) Uniform on $[0, 2]$, and (c) Bimodal on $\{0.1, 9.1\}$ with probabilities $\{0.9, 0.1\}$. Both algorithms Algo2 and Algo4 start with the same random initial matching and subsequently they are provided with the same random matchings. Both the algorithms run till they obtain a matching whose weight is at least a pre-determined fraction f of the weight of MWM on the same graph. The average number of iterations taken by an algorithm to achieve this weight is used as a measure of its learning time. When an algorithm takes more than 10000 iterations to learn this weight, we simply report the number of iterations as 10000.

Results: For each $f \in [0.1, 0.9]$, and for each distributions, we obtain the average number of iterations over 100 sample runs. The results are plotted in the Figure 3.6. The X-axis is the fraction f while Y-axis the average number of iterations taken to learn the fraction f by algorithm. "Uni" represent uniform, "Bi" represents Bimodal and "Exp" represents Exponential. Results show that for all distributions, both algorithms manage to learn quickly when $f \leq 0.2$. But as f grows the average number of iterations taken by Algo2 is very high

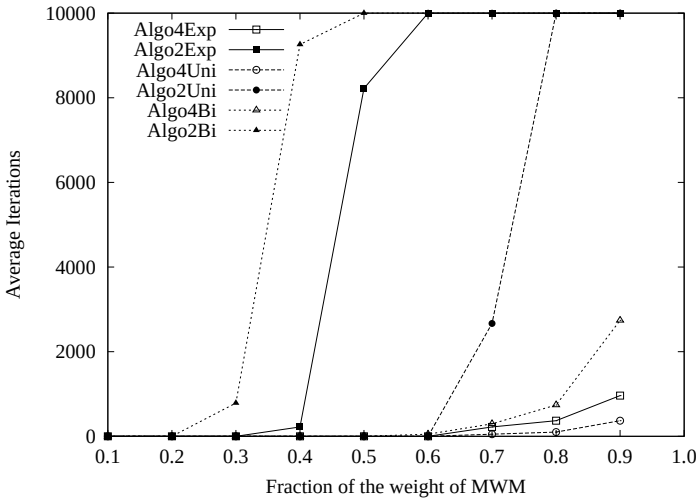

Figure 3.6: Learning time: Algo2 v/s Algo4.

compared that of Algo4. We also note that learning time gets worse as the variance of the edge-weight distribution increases; i.e. Uniform is easier to learn than Exponential distribution which is easier to learn than Bimodal distribution.

Comparison of Algo2 *and* Algo4*: Analytical Model*

Analytical Model: The simulation study showed that Algo4 learns "good" matching a lot quicker compared to Algo2 under different edge-weight distributions. It is not easy to obtain such qualitative result analytically for any general edge-weight distribution. As our interest is in determining learning time of algorithm for MWM, we consider a simplified model in which the edges of MWM are assigned weight ∞(a large enough value) and all other edges are assigned weight 0. Without loss of generality we assume that the MWM is the identity matching. Thus the edge-weight matrix of the bipartite graph has ∞ on n entries of the main diagonal and 0 in remaining $n^2 - n$ positions. We compare the performance of the Algo2 and Algo4 in this context. Each time both algorithms are provided the same random matching. The MWM is learned when all edges of the identity matching are learned by the algorithm.

Performance of Algo2: First observe that the matching retained by Algo2 at the end of time m is the matching with the most of edges in common with the identity matching, among all random matchings chosen till time m. An edge i of a matching is said to be *fixed* if input i is matched to output i. Note that all the elements of the identity matching are fixed. The

learning time of an algorithm in this context is simply the time taken to learn the edges of the identity permutation. Therefore, the probability distribution of the number of fixed edges in a randomly chosen permutation can given us the distribution of the fixed elements learnt by Algo2 till time m. This distribution is well-studied in the literature in various contexts. We prove some required results about this distribution for the sake of completeness.

Let A_i denote the event that i^{th} element is fixed in a randomly chosen permutation π^r. Let P_k^n denote the probability that exactly k elements are fixed in π^r. First, let us compute P_0^n, the probability that no element is fixed.

$$\begin{aligned} P_0^n &= \Pr(\cap_{i=1}^n A_i^c) \\ &= 1 - \Pr(\cup_{i=1}^n A_i) \\ &\stackrel{(a)}{=} 1 - \sum_{j=1}^{n} (-1)^{j+1} \binom{n}{j} \frac{(n-j)!}{n!} \\ &= \sum_{j=0}^{n} (-1)^j \frac{1}{j!} \approx e^{-1}, \end{aligned}$$

where (a) is a direct application of the Inclusion-Exclusion principle. In general, for all k,

$$\begin{aligned} P_k^n &= \frac{\binom{n}{k} P_0^{n-k} (n-k)!}{n!} \\ &= O\left(\frac{1}{k!}\right). \end{aligned}$$

Hence, Q_k^n, the probability that a randomly chosen permutation has at least k fixed elements, is given as

$$\begin{aligned} Q_k^n &= \sum_{j \geq k} P_j^n \\ &= O\left(\sum_{j \geq k} \frac{1}{j!}\right) \\ &= O\left(\frac{1}{k!}\right). \end{aligned} \tag{3.19}$$

This leads to the following Lemma.

Lemma 1. *The algorithm* Algo2 *takes* $\Theta(k!)$ *time to learn* k *fixed elements.*

Performance of Algo4: Now, we consider Algo4. We will show that the order of the learning time for Algo4 is significantly smaller than that of Algo2. Recall that under algorithm Algo4, matching $\pi(m+1)$ at time $m+1$ is obtained by merging a random matching $\pi^r(m+1)$ chosen at time $m+1$ with $\pi(m)$ chosen at time m. The Merge procedure considers cycles with edges alternatively belonging to $\pi(m)$ and $\pi^r(m+1)$. This is the same as considering the cyclic decomposition of a random permutation. Now, for each cycle Merge procedure either picks all edges from $\pi(m)$ or all edges from $\pi^r(m+1)$. Hence it is important to know the distribution of cycles in a random permutation. This distribution is well-studied. Let $K(m)$ be the random number of cycles induced by the cyclic decomposition of $\pi(m)$ and $\pi^r(m+1)$ and let $C_l(m), 1 \leq l \leq K(m)$ be the length of the l^{th} cycle. Let us remind ourselves that $K(m)$ and $C_l(m), 1 \leq l \leq K(m)$ are IID random variables across time m. Now, it is well-known that $K(m)$ is sharply concentrated around its mean $\log_e n$. Though the distribution of cycle-lengths $C_l(m)$ is not concentrated around its mean $n/\log_e n$, for simplicity we assume the following: there are $\log_e n$ cycles each of length $n/\log_e n$. It can be shown that this assumption gives weaker upper bound on learning time of Algo4. Next we compute the bound on iterations taken by Algo4 to learn almost all elements of MWM in this context.

Let $X(m)$ be the number of fixed elements in $\pi(m)$, that is elements of MWM already learnt by $\pi(m)$ by time m. We would like to lower bound the probability of the event that the number of fixed elements will increase in $\pi(m+1)$ given $X(m)$. Consider the following event: $\pi^r(m+1)$ contains a fixed element and it belongs to a cycle which does not contain any of the $X(m)$ fixed elements of $\pi(m)$. In this case the Algo4 will pick elements of $\pi^r(m+1)$ for this cycle. This in turn increases the number of fixed elements in $\pi(m+1)$ to at least $X(m)+1$. Hence whenever this event happens the number of fixed elements of $\pi(m+1)$ increases. Next we compute the probability of this event.

The probability that there are k fixed elements in $\pi^r(m+1)$ is $O(1/k!)$ as computed above. The $X(m)$ fixed elements of $\pi(m)$ are distributed among $\log_e n$ cycles uniformly at random. A cycle contains $n/\log_e n$ elements of $\pi^r(m+1)$ and $\pi(m)$ each. Consider one of the fixed element of $\pi^r(m+1)$. Now, the probability that the cycle containing a fixed element

of $\pi^r(m+1)$ does not contain any of $X(m)$ element is:

$$\begin{aligned} p &= \frac{\binom{n-X(m)}{n/\log_e n}}{\binom{n}{n/\log_e n}} \\ &\approx \left(1-\frac{X(m)}{n}\right)^{\frac{n}{\log_e n}} \end{aligned} \tag{3.20}$$

From the above discussion, on average the increase in $X(m+1)$ from $X(m)$ is lower bounded as:

$$\begin{aligned} E[X(m+1)-X(m)] &\geq \sum_{k\geq 1}\frac{1}{k!}kp \\ &= \sum_{k\geq 0}\frac{1}{k!}p \\ &\approx (1-X(m)/n)^{n/\log n} \\ &\approx \exp\left\{-\frac{X(m)}{\log n}\right\}. \end{aligned} \tag{3.21}$$

Let $y(s)=E[X(sn)]/n$. Then, we obtain the following differential equation for large n:

$$\frac{dy(s)}{ds}=\exp\left\{-\frac{y(s)}{\log n}\right\}.$$

The solution of this equation is given by

$$\frac{\log n}{n}\left(\exp\left\{\frac{ny(s)}{\log n}\right\}-1\right)=s. \tag{3.22}$$

From (3.22), we obtain

$$\log n\left(\exp\left\{\frac{X(m)}{\log n}\right\}-1\right)=m. \tag{3.23}$$

The (3.23) leads to the following Lemma.

Lemma 2. *The algorithm* Algo4 *takes* $\Omega\left(\log n\left(\exp\left\{\frac{k}{\log n}\right\}-1\right)\right)$ *time to learn* k *fixed elements.*

Comparison of Algo2 *and* Algo4 *Under Analytic Model*: Let $T_2(n)$ and $T_4(n)$ denote the average time it takes for Algo2 and Algo4 to learn n fixed elements under the above described analytic model. Then, results of Lemma 1 and Lemma 2 imply that $T_2(n)=\Theta(n!)$ while

$T_4(n) = \Theta(\log n \exp(n/\log n))$, which yields the following Theorem.

Theorem 9. *Under the simple analytical model described above, the average time it takes for* Algo2 *and* Algo4 *for learn MWM, denoted by* $T_2(n)$ *and* $T_4(n)$ *respectively, are related as*

$$T_2(n) \quad = \quad \Omega\left(T_4(n)^{\log^2 n}\right). \tag{3.24}$$

Theorem 9 indicates the drastic difference between the learning time of the algorithms using Merge and Max. This in turn provides credibility to the Merge procedure.

3.2.3 SERENA: Use of Arrival Information

The SERENA algorithm can be seen as a variant of LAURA in the sense that it does not use external non-uniform random sampling procedure. Instead it uses arrival information to obtain a new matching every time. In summary, SERENA has the following three features: (1) use of memory, (2) use of arrival information to obtain new matching, and (3) Merge procedure.

SENENA Algorithm

Now, we describe the algorithm. As before, let $\pi(m)$ be the matching used by SERENA at time m. Recall that $A(m+1) = [A_{ij}(m+1)]$ is the arrival matrix at time $m+1$. That is, $A_{ij}(m+1) = 1$ denotes that a packet arrived at input i for output j at time $m+1$. The algorithm finds schedule at time $m+1$ as follows.

(a) Compute matching $\pi^A(m+1)$ by applying procedure Arr-Matching on arrival matrix $A(m+1)$, which uses queue-size matrix $Q(m+1)$.

(b) The schedule at time $m+1$ is $\pi(m+1) = \text{Merge}(\pi(m), \pi^A(m+1))$.

ARR-MATCHING Procedure

The procedure Arr-Matching obtains a matching from a given arrival matrix $A = [A_{ij}]$. By definition, A is such that each of the $A_{i\cdot} \in \{0, 1\}$ for all i, that is, each input can have at most one arrival. But, more than one packets (possibly n in the worst case) can arrive for the same output, that is, $A_{\cdot j} \in \{0, \ldots, n\}$ for all j. This structure suggests one possible simple

way to obtain good matching is as follows: (1) if there is no j such that $A_{\cdot j} > 1$, then A is a possibly sub-matching. Let π^A be this matching. Connect the inputs and outputs that are not matched under π^A in any order. This yields the matching; (2) otherwise, there are $j_1, \ldots, j_k, k \leq n/2$ such that $A_{\cdot j_l} \geq 2, l = 1, \ldots, k$. Now we create π^A as follows. Initially, set $\pi^A = [0]$. For all $l = 1, \ldots, k$, do the following: *Set* $\pi^A_{i_l j_l} = 1$, *where*

$$i_l = \arg \max_{1 \leq i \leq n} \{Q_{ij_l} A_{ij_l}\}.$$

In the π^A, thus obtained, will have some inputs and outputs such that they are not connected (i.e. corresponding rows and columns do not have any entry equal to 1). Connect these in an arbitrary fashion and obtain a complete matching. This completes the description of Arr-Matching. The Figure 3.7 explains how the Arr-Matching for a particular example.

Properties of SERENA

The following theorem states the throughput and delay property of SERENA.

Theorem 10. *The algorithm* SERENA *is stable under Bernoulli IID arrival process with admissible arrival rate-matrix* λ. *Further, the average queue-size is bounded as*

$$\sum_{i,j} E[Q_{ij}] \quad \leq \quad \frac{n(n+\Lambda)}{1-\lambda^*}, \tag{3.25}$$

where $\Lambda = \left(\frac{1}{\lambda_*(1-\lambda^*)}\right)^n$.

Proof. We will show that under Bernoulli IID traffic with admissible arrival rate-matrix λ, the

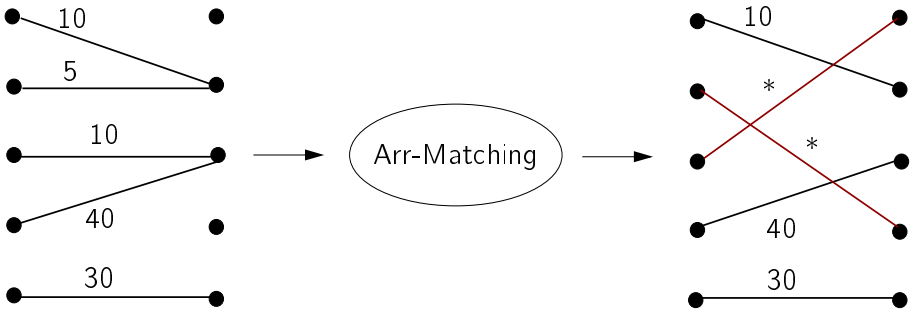

Figure 3.7: An example of Arr-Matching procedure.

algorithm SERENA is $(\Lambda,1)$-MWM. Then, a direct application of Theorem 2 of Chapter 2 will imply the statement of Theorem 10.

Now, we will show that SERENA is $(\Lambda,1)$-MWM. If we show that the probability of SERENA picking MWM as schedule at any given time is lower bounded by $1/\Lambda$, then by the arguments used in the proof of Algo2, we immediately obtain that SERENA is $(\Lambda,1)$-MWM. Hence, we are left will showing that the probability of SERENA picking MWM as schedule at any given time is lower bounded by $1/\Lambda$.

Consider any time m. Let $\pi^*(m)$ be the maximum weight matching at this time m. We wish to lower bound the probability of the event that SERENA uses $\pi^*(m)$ as its schedule. Now, consider $\pi^*(m)$. Let there be k,$0 \leq k \leq n$, input-output pairs that are matched under $\pi^*(m)$ such that the edges between them have arrival rate 0. If there are none such input-output pair then neglect them in the remainder of the discussion. Without loss of generality, let these inputs and outputs be numbered $1,\ldots,k$. Now consider the following event: (1) no packets arrive at inputs $1,\ldots,k$, and (2) packets arrive at inputs $k+1,\ldots,n$ precisely for the outputs that are connected by $\pi^*(m)$. This event will imply that Arr-Matching will produce a matching that is maximum weight matching, and hence SERENA will use maximum weight matching as a schedule. Now the probability of (1) is at least $(1-\lambda^*)^k$ and probability of (2) is $(\lambda_*)^{n-k}$ Since $0 \leq k \leq n$, this probability is strictly larger than $(\lambda_*\lambda^*)^n$. Thus, we showed that SERENA uses maximum weight matching as its schedule with probability $1/\Lambda$ as desired. This completes the proof of Theorem 10. □

Simulation Results

We study the performance of SERENA via extensive simulation study. The simulation setup is identical to the one described in Section 3.1.3 of this chapter. We compare the performance of SERENA with the MWM, iSLIP and iLQF algorithms under the Diagonal traffic. The results are shown in Figure 3.8. The algorithms SERENA and MaxSERENA (the maximized version of SERENA) perform quite competitively with respect to MWM.

3.2.4 Implementation

In this section, we discuss implementation of all the three algorithms – APSARA, LAURA and SERENA.

APSARA: All versions of APSARA involve a Hamiltonian walk. As noted in section 3.1,

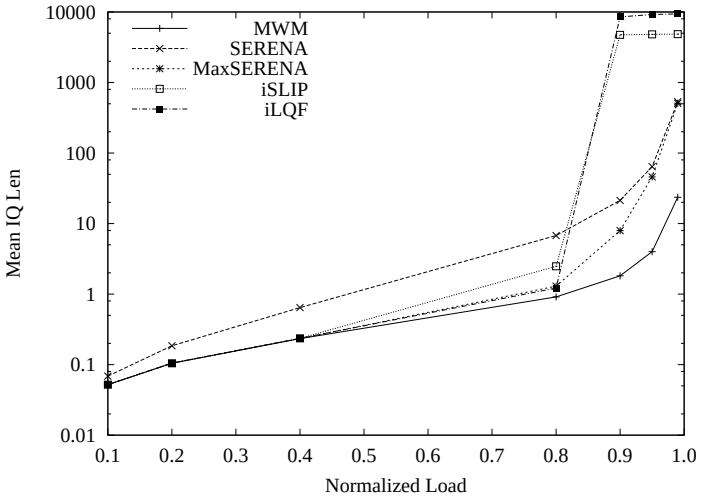

Figure 3.8: Performance of APSARA under Diagonal traffic.

finding next permutation in the Hamiltonian walk requires constant number of operations. Moreover, we find that practically, we do not need Hamiltonian walk. All simulation results remain unchanged when we ignore Hamiltonian walk. Thus, Hamiltonian walk is a purely theoretical tool used in Theorems to provide stability. Thus, while the walk is extremely simple to implement, we do not consider it either in implementation or in performance evaluation The main feature of APSARA is that it can be implemented in a parallel architecture very efficiently. Figure 3.9 shows a schematic for the implementation of APSARA with K modules. As shown in the Figure 3.9, the old matching $\pi(m)$ and the new arrivals, $A(m+1)$, are used to compute the weights of the K neighbor matchings in parallel. Arrival information is required to update at most n queues. Computing weight of each neighbor involves 2 additions and 2 subtractions. The new matching, $\pi(m+1)$ is the one with highest weight among all the K neighbors and the $\pi(m)$. Computing the maximum can be done in $\log K$ time with $O(K)$ hardware space. The above computation can be easily pipelined as a loss of very little performance.

LAURA: There are two tasks performed in LAURA: (1) Non-uniform random sampling and (2) Merge procedure. Under non-uniform random sampling, I random permutations are chosen, each of which may require $O(n \log n)$ coin-flips (or equivalent computation). The standard parameter setting is such that $I = \log_2 n$. Hence, the operations involved in non-uniform random sampling is $O(n \log_2^2 n)$. Two, Merge procedure. The Merge procedure essentially

involves n addition and n subtractions. Thus, it takes precisely $2n$ operation. Hence, net amount of work in LAURA is $O(n \log_2^2 n) + 2n$. We note that, if cost of randomness is ignored then non-uniform sampling will require computation of $O(nI) = O(n \log n)$ operations just to select useful edges. Though, algorithm LAURA is simple, it does not seem simple enough. Certainly, it can be very useful algorithm given enough resources.

SERENA: The algorithm SERENA performs two main tasks: (1) Arr-Matching procedure and (2) Merge procedure. As discussed above, the Merge procedure requires precisely n additions and n subtractions. The Arr-Matching is required to resolve conflicts between edges at output side. This requires at most n comparisons. The Arr-Matching is required to match unmatched input-output nodes. This is done in any arbitrary fashion. A round-robin algorithm (or algorithms like Wave Front Arbiter) requires $O(n)$ operations. Thus, SERENA algorithm is truly very simple and does not require any external randomization.

3.2.5 Simulation Under Correlated Traffic

The algorithms – APSARA, LAURA and SERENA – try to learn the weight of the MWM schedule using different techniques. Depending on the arrival process, the rate at which algorithms can learn the weight may change, which in turn may change their performance. The simulation study in Section 3.2 was based on friendly, completely independent (and hence no correlation) Bernoulli IID arrival process. We study the change in performance of algorithms when there is a strong correlation in arrival process. Intuitively, temporal correlation in traffic could help an algorithm to learn quicker and achieve better performance. We verify our intuition with simulation results.

The simulation setup is identical to the one described in Section 3.1.3 except that arrival process is correlated. We describe the model to generate correlated arrival process. The traffic is generated according to correlated "bursty" traffic, with burst parameter B. Let λ be the arrival rate-matrix. The cell arrival process at each input i is characterized by a two-state ON-OFF model. Every input has its own two state (ON and OFF) Markov chain. At any time slot, input i jumps from ON to OFF state with probability $\left(\frac{1}{B\lambda_{i\cdot}}\right)$ and jumps from OFF to ON state with probability $\left(\frac{1}{B(1-\lambda_{i\cdot})}\right)$. When i is in OFF state, it does not generate any packet. When i is in ON state it generates packets. Now, when i is in ON state for contiguous time slots, it generates packets only for one output, which is chosen to be j with probability $\lambda_{ij}/\lambda_{i\cdot}$ at random when i enters ON state from OFF state.

Under the above simulation setup, the Figure 3.10 shows the mean queue-length of algorithms of interest as a function of the average burst-size, B. Note that, $B = 1$ correspond to the results of Bernoulli IID traffic, which is plotted in Figure 3.12.

All the three proposed algorithms behave closer to the MWM as the average burst size (i.e. the degree of correlation in the traffic) increases. Correlation can indeed help, since the correlation among subsequent maximum weight matchings is captured by the memory retained in the previous matching.

3.3 Chapter Summary and Discussion

The results of the previous chapter suggested that MWM (and its approximations) perform very well. But, it is not feasible to implement the MWM or its known approximations. This motivated us to seek implementable approximation of MWM that perform very well. In this chapter, we present novel design approaches to obtain simple-to-implement approximation algorithms of MWM. We exploited the following general features of the switch scheduling problem in designing such algorithms: (i) the use of memory, (ii) the randomized weight augmentation, (iii) the randomness and the information provided by recent arrivals, and (iv) parallelism that naturally arise due to structure of the space of permutations.

We developed three algorithms – APSARA, LAURA and SERENA – to exploit the above-mentioned features. We analyzed their throughput and delay properties theoretically and found that they are all stable. An extensive simulation study demonstrated that the algorithms approximate the performance of MWM very well. We clearly spelled out the implementation details of these algorithms. We strongly believe that these algorithms are implementable in the current core-routers at a very little implementation cost.

The design methods of this chapter are quite general. They can be applied for a large class of problems in networking (and other systems setting) where the "continuity of state" is observed.

The results of this chapter show that it is not very difficult to obtain an approximation algorithm of the MWM in a stochastic setting, that is, when the weights of underlying bipartite graph are changing by little every time in a stochastic manner. An interesting theoretical question that arises from this work is as follows: what is the inherest complexity of finding the Maximum Weight Matching exactly (not approximately) when little change happen in the weight of bipartite graphs. Is it still $O(n^3)$ or can we do really better. We believe that when

all the weights are distinct most of the time, then the complexity can be significantly reduce.

3.4 Bibliographic Notes

The basic randomized algorithm, Algo2 was first proposed by *Tassiulas and Ephremides* [1992]. The proof of stability of Algo2 presented in this chapter, is quite different from the proof by *Tassiulas and Ephremides* [1992]. The algorithms – APSARA, LAURA and SERENA – were published by *Giaccone et al.* [2003]. The related work to the results of this chapter can be found in the works by *Shah et al.* [2001], *Giaccone et al.* [2001] and *Giaccone et al.* [2002].

A lot of work has been done in the past to obtain simple-to-implement high performance scheduling algorithms and in particular approximations to MWM. Among commercially available routers, mainly variants of the three maximal type matching algorithms are implemented. These three algorithms are iSLIP, Parallel Iterative Matching (PIM) and Wave Front Arbiter (WFA). The iSLIP algorithm was proposed by *McKeown* [1995]. A detailed exposition on iSLIP and its variants can be found in work by *McKeown* [1999]. The PIM algorithm was proposed by *Anderson et al.* [1993]. It originated during the design of AN2 switches of former DEC. The WFA algorithm was proposed by *Tamir and Chi* [1993]. These algorithms, though very simple to implement, are rather poor in performance. For example, simulation results presented in this chapter exposes the weakness of iSLIP algorithm.

Among other known approximations, the greedy Maximum Weight Matching algorithm, also called iLQF was formally studied by *McKeown* [1995]. The iLQF algorithm is $(0,0.5)$-MWM. It provides lower throughput and high delay (see simulations of this chapter). Many other algorithms have been proposed, for example RPA was proposed by *Marsan et al.* [1999], MUCS by *H.Duan et al.* [1997] etc. None of these algorithm compare well with the algorithm proposed in this chapter.

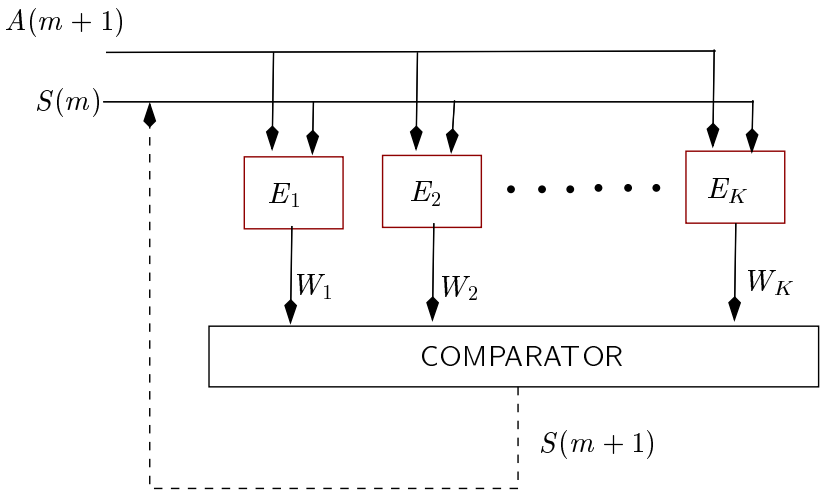

Figure 3.9: A schematic for the implementation of APSARA.

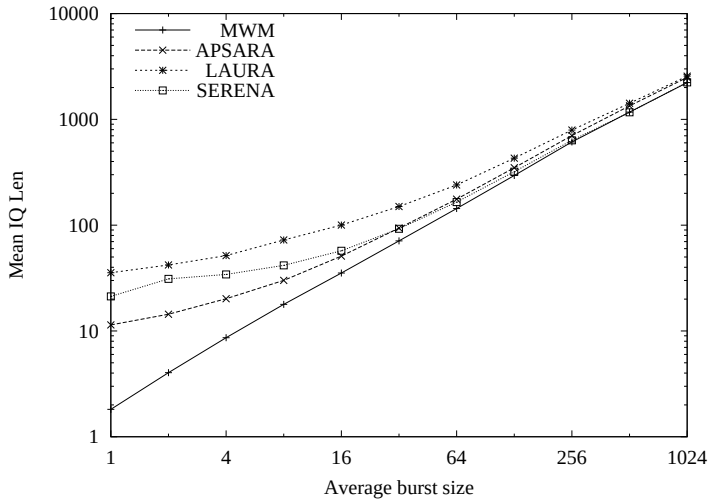

Figure 3.10: Performance under ON/OFF traffic with input load 0.9.

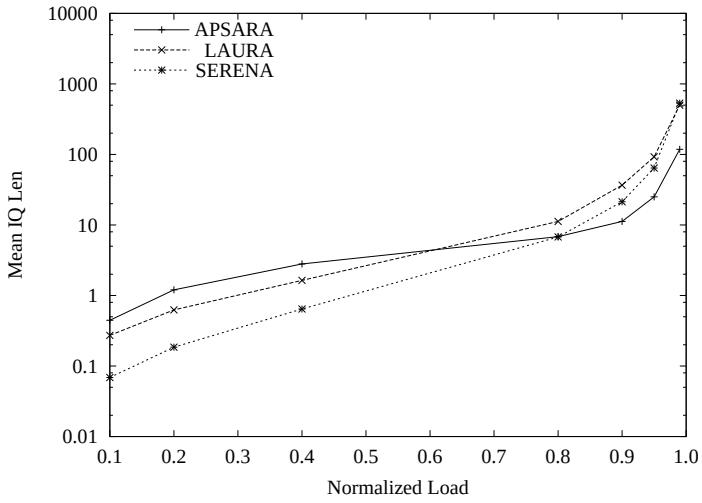

Figure 3.11: Comparison of APSARA, LAURA and SERENA: Diagonal traffic.

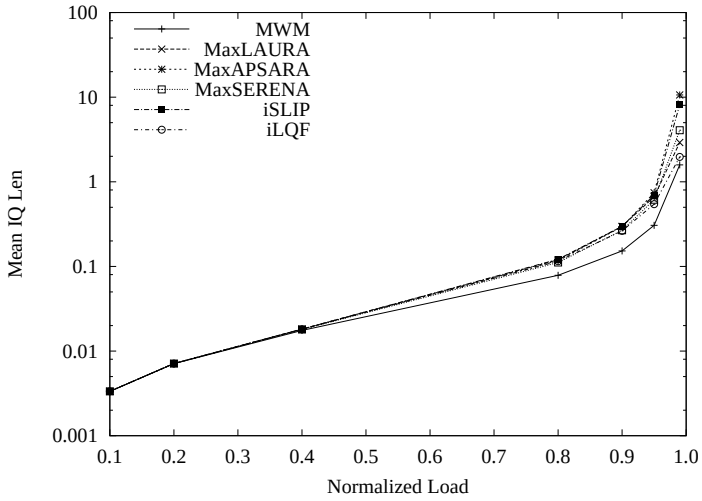

Figure 3.12: Comparison of APSARA, LAURA and SERENA: Uniform traffic

CHAPTER 4

Fluid Models, Heavy Traffic and Delay

In the previous two chapters, we studied Maximum Weight Matching and its approximation algorithms. The Chapter 2 showed that under Bernoulli IID arrival process, MWM and its approximations have good throughput and delay properties. The implementation concerns motivated design of simple approximations to MWM that have comparable performance to MWM. The Chapter 3 presented various design methods to obtain approximation to MWM. These methods are general enough in the sense of they can be used to design good approximation to MWM even when weight are different from queue-sizes. This naturally leads to the following questions: (1) what are all possible weight for which MWM stable? (2) among all such stable MWM, what weight selection minimizes the average queue-size or delay?

In this chapter, we present answers to both of the above questions. To answer the first question, we define a general class of MWM algorithm, denoted by MWMf. The MWMf algorithm chooses maximum weight matching as schedule, where weight of an edge (i,j) is $f(Q_{ij})$ instead of queue-size Q_{ij} for some real valued function f. As a special case, when $f(x) = x$, MWMf becomes the usual MWM. We characterize the class of functions f under which MWMf is stable. To answer the second question, we study MWMf under the special class of functions parametrized by $\alpha \in \mathbb{R}_+$. The function corresponding to parameter α is $f(x) = x^\alpha$. Again, for $\alpha = 1$, the algorithm corresponds to MWM. The algorithm corresponding

to parameter α is denoted by MWM-α. We characterize an optimal algorithm among all possible scheduling algorithm (not only among MWMf algorithms) as the limiting algorithm, MWM-0^+, which is obtained as $\alpha \to 0^+$. The MWM-0^+ is a Maximum Weighted Maximum Size Matching. That is, among all Maximum Size Matchings, it serves the one that has maximum weight (weight is logarithm of queue-size). We also find that the usual MWM is not optimal and thus contradicting the long standing folk-lore in the switching community about the optimality of MWM. We use similar methods to resolve the Conjecture 1 stated in Chapter 1.

To obtain the above claimed answers we use fluid model technique and heavy traffic analysis of switches. We first present useful definition and notation for this chapter in Section 4.1. We present formal fluid model of a switch in Section 4.2. Then, we use the fluid model to prove stability MWMf class of algorithms. In Section 4.3 we study fluid models of switch under MWMf algorithm when switch is loaded critically, that is, one or more of n inputs and n outputs are loaded to its capacity. This section requires us to prove combinatorial properties of MWMf algorithms in order to characterize the space of fixed points of critical fluid model. The results of 4.3 are of particular interest as they are essential in obtain the behavior of system under heavy traffic. In Section 4.4, we introduce the set up of heavy traffic scaling for a switch. We define and characterize the "state space collapse" space of a switch using results of Section 4.3. Using the state space collapse characterization of MWMf algorithm and in particular MWM-α, we obtain the optimality of MWM-0^+ under heavy traffic in Section 4.5. Further, we present explanation for Conjecture 1. Finally, Section 4.6 presents discussion, scope of the method developed in this chapter and chapter summary.

4.1 Preliminaries

The MWM algorithm chooses schedule $\pi^*(m)$ as a schedule at time m, where $\pi^*(m)$ is such that it satisfies

$$\pi^*(m) \quad = \quad \arg\max_{\pi \in \mathbb{P}} \{\pi \cdot Q(m)\}. \tag{4.1}$$

Equivalently, (4.1) can be interpreted in terms of cumulative service vector $(S_\pi(m))_{\pi \in \mathbb{P}}$ as follows:

$$S_\pi(m) > 0 \quad \Rightarrow \quad \sigma \cdot Q(m) \leq \pi \cdot Q(m), \quad \forall\, \sigma \in \mathbb{P}. \tag{4.2}$$

Now we define a generalized Maximum Weight Matching algorithm. Consider any function $f : \mathbb{R}_+ \to \mathbb{R}_+$. Then, MWMf algorithm chooses the schedule so that the following is always satisfied:

$$S_\pi(m) > 0 \quad \Rightarrow \quad \sigma \cdot f(Q(m)) \leq \pi \cdot f(Q(m)), \quad \forall\, \sigma \in \mathbb{P}, \tag{4.3}$$

where recall that $f(Q(m)) = (f(Q_{ij}(m)))_{ij}$. Thus, at time m MWMf chooses a maximum weight matching as the schedule with weight of edge (i,j) as $f(Q_{ij}(m))$. In this chapter, we consider functions f that satisfy the following condition.

Condition 1. *The function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is a strictly increasing continuous function with $f(0) = 0$. Further, for any $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ in $\mathbb{R}_+^n$*

$$\sum_i f(x_i) \geq \sum_i f(y_i) \quad \textit{implies} \quad \sum_i f(\delta x_i) \geq \sum_i f(\delta y_i), \quad \forall \;\; \delta > 0. \tag{4.4}$$

Some examples of function that satisfy condition 1 are $f(x) = x^2$, $f(x) = \log x$ etc. A special sub-class of MWMf algorithms of our interest are the class of algorithms parametrized by $\alpha \in \mathbb{R}_+$ and denoted by MWM-α algorithms. An MWM-α uses $f(x) = x^\alpha$ as its weight function.

4.2 Fluid Model and Stability of MWMf

In this section we first introduce the fluid model of a switch. We described what we mean by fluid model of a switch. For the ease of understanding, the formal description of fluid model associated with the switch is presented first. Then, we provide justification. Finally, we use fluid model to prove the stability of MWMf algorithm.

4.2.1 Fluid Model of a Switch

As described in chapter 1, the dynamics of a switch can be described completely by the tuple $\mathcal{X}(m) = (\bar{A}(m), D(m), Q(m), S(m)), \quad m \in \mathcal{Z}_+$. The fluid scaling of switch is developed essentially to study the behavior of system at the "rate" level. Under the fluid scaling, instead of looking at $\mathcal{X}(m)$, the focus is in studying the behavior of the tuple

$x(t) = (a(t), d(t), q(t), s(t)), t \in \mathbb{R}_+$, where

$$x(t) = \lim_{r\to\infty} \frac{\mathcal{X}(rt)}{r}, \tag{4.5}$$

where

$$\mathcal{X}(t) = (1 - t + \lfloor t \rfloor)\mathcal{X}(\lfloor t \rfloor) + (t - \lfloor t \rfloor)\mathcal{X}(\lfloor t \rfloor + 1).$$

Note that, as defined above, $x(t)$ represents one of possibly many limit points. One does not require existence of a unique fixed point, as generally one proves properties for all limit points.

From Assumption 1, inter-arrival times are IID. Hence by Strong Law of Large Numbers (or ergodic theorem) for IID variable,

$$\lim_{m\to\infty} \frac{1}{m}\bar{A}(m) = \lambda, \quad \text{with probability} \quad 1. \tag{4.6}$$

This in turn implies that, with probability 1,

$$a(t) = \lambda t. \tag{4.7}$$

Now, we are ready to describe the dynamics of the switch and the corresponding fluid model equations. The fluid model equations are essentially the equations that govern the evolution of quantities of $x(t)$. Though, these fluid model equations are intuitively clear, the formal justification is not straightforward. The formal justification is given later. Similar to the treatment in Chapter 1, we consider the dynamics of a switch in two separate components: (i) Algorithm-independent dynamics, and (ii) Algorithm-dependent dynamics.

Algorithm-independent Dynamics

The quantities $Q(\cdot), \bar{A}(\cdot)$ and $D(\cdot)$ are related by the following basic queueing equation.

$$\begin{aligned} Q(m) &= Q(0) + \bar{A}(m) - D(m) \\ &= \bar{A}(m) - D(m), \end{aligned} \tag{4.8}$$

since $Q(0) = 0$ from Assumption 2. In each time slot, at most one of the permutation is served and we are interested in non-idling switches. Hence,

$$\sum_{\pi \in \mathbb{P}} S_\pi(m) \quad = \quad m. \tag{4.9}$$

Clearly, $D(m)$ and $\{S_\pi,\ \pi \in \mathbb{P}\}$ are related to each other. Specifically,

$$D_{ij}(m) \quad = \quad \sum_{\pi \in \mathbb{P}} \sum_{\ell=1}^{m} \pi_{ij} \mathbf{1}_{Q_{ij}(\ell)>0} \left(S_\pi(\ell) - S_\pi(\ell-1) \right), \quad \forall\ i,\ j. \tag{4.10}$$

Equivalently,

$$D_{ij}(m) - D_{ij}(m-1) \quad = \quad \sum_{\pi \in \mathbb{P}} \pi_{ij} \mathbf{1}_{Q_{ij}(m)>0} \left(S_\pi(m) - S_\pi(m-1) \right), \quad \forall\ i,\ j. \tag{4.11}$$

Next, we describe the corresponding dynamics of fluid scaled switch. The basic fluid quantities $(q(t), d(t), a(t), s(t))$ are absolutely and hence differentiable almost everywhere w.r.t. the Lebesgue measure. We will talk about fluid model equations as the differentiable t. The equations analogous to (4.8)-(4.10) are the following.

$$q(t) \quad = \quad \lambda t - d(t), \tag{4.12}$$

$$\sum_{\pi \in \mathbb{P}} s_\pi(t) \quad = \quad t, \tag{4.13}$$

$$\dot{d}_{ij}(t) \quad = \quad \sum_{\pi \in \mathbb{P}} \pi_{ij} \mathbf{1}_{q_{ij}(t)>0} \dot{s}_\pi(t). \tag{4.14}$$

We define additional notation, which will be useful in further exposition. The $n \times n$ matrix of instantaneous service rates, $\sigma(t)$, is

$$\sigma(t) \quad = \quad \sum_{\pi \in \mathbb{P}} \pi \dot{s}_\pi(t). \tag{4.15}$$

Then the equations (4.12) and (4.14) can be re-written as follows:

$$\dot{q}_{ij}(t) = \begin{cases} \lambda_{ij} - \sigma_{ij}(t) & \text{if } q_{ij} > 0 \\ \left(\lambda_{ij} - \sigma_{ij}(t) \right)^+ & \text{otherwise} \end{cases} \tag{4.16}$$

The above equations can be written in the following compact form.

$$\dot{q}(t) = \big(\lambda - \sigma(t)\big)^{+[q=0]}. \tag{4.17}$$

Algorithm-dependent Dynamics

The above are the basic equations which govern a switch, regardless of the scheduling algorithm. For a specific scheduling algorithm there may be additional equations. The algorithm decides which permutations are chosen for service, that is, $\{S_\pi(\cdot), \pi \in \mathbb{P}\}$. We describe the dynamics for MWMf algorithm. Under the MWMf algorithm, the equation (4.3) is satisfied, which is the following.

$$S_\pi(m+1) = S_\pi(m) \quad \text{if} \quad \pi \cdot f(Q(m)) < \max_{\rho \in \mathbb{P}} \rho \cdot f(Q(m)), \quad m \in \mathcal{Z}_+. \tag{4.18}$$

The corresponding fluid model equation is given by

$$\dot{s}_\pi = 0 \quad \text{if} \quad \pi \cdot f(q) < \max_{\rho \in \mathbb{P}} \rho \cdot f(q). \tag{4.19}$$

4.2.2 Justification of Fluid Model

In this section, we present justification of fluid model equations (4.12)-(4.14) and (4.19). We first introduce some definitions and notations. We want to study the limiting quantity $x(t)$, where

$$x(t) \quad = \quad \lim_{r \to \infty} \frac{\mathcal{X}(rt)}{r}, \tag{4.20}$$

where

$$\mathcal{X}(t) = (1 - t + \lfloor t \rfloor)\mathcal{X}(\lfloor t \rfloor) + (t - \lfloor t \rfloor)\mathcal{X}(\lfloor t \rfloor + 1).$$

Equivalently, we wish to study $\lim_{r\to\infty} x^r(t)$ where, for $r \in \mathbb{R}_+$

$$x^r(t) = \mathcal{X}(rt)/r.$$

In particular, we wish to study $x(t)$ over a time interval $[0, T]$, where $T \in \mathbb{R}_+$ a finite constant. Each $x^r(t)$ has associated probability measure $\mu^r(\cdot)$. Our interest is in studying the limiting measure $\mu(\cdot)$ obtained as the limit of $\mu^r(\cdot)$ as $r \to \infty$. Basic questions are: what is the space

on which $\mu^r(\cdot)$ is defined? are limit points of $\mu^r(\cdot)$ probability measures? if yes, can we obtain their characterization ? Answering these questions is equivalent to showing that the limits of $x^r(\cdot)$ satisfy fluid model equations.

Now we proceed towards studying limits of $\mu^r(\cdot)$. The space of $\mu^r(\cdot)$ is the set of all values taken by $x^r(t)$ as $r \to \infty$ over a finite time $[0,T]$. For this we need following definition. A sequence of functions $\{f^r,\ r \in \mathbb{R}_+\}$ where $f^r : [0,T] \to \mathbb{R}_+$, is said to converge uniformly on compact intervals (u.o.c.) to a function $f : [0,T] \to \mathbb{R}_+$ if

$$\lim_{r\to\infty} |f^r - f|_T = 0,$$

where the $|f^r - f|_T$ is the sup-norm defined as

$$|f^r - f|_T = \sup_{0\le t\le T} |f^r(t) - f(t)|.$$

Next, we note the following properties of $\mathcal{X}(\cdot), x^r(\cdot)$ which prove their Lipschitz continuity.

1. At most one packet can arrive at an input in a time slot. Hence, for all i,j

$$|A_{ij}(m+\ell) - A_{ij}(m)| \le \ell,\ \forall m,\ell \quad \Rightarrow \quad |a^r_{ij}(t+s) - a^r_{ij}(t)| \le s,\ \forall t,s. \tag{4.21}$$

2. At most one packet can depart from an input (as well as output) in a given time slot. Hence, for all i,j

$$|D_{ij}(m+\ell) - D_{ij}(m)| \le \ell,\ \forall m,\ell \quad \Rightarrow \quad |d^r_{ij}(t+s) - d^r_{ij}(t)| \le s,\ \forall t,s. \tag{4.22}$$

3. Everytime one of the matching is scheduled to transfer unit packet. Thus, for all $\pi \in \mathbb{P}$

$$|S_\pi(m+\ell) - S_\pi(m)| \le \ell,\ \forall m,\ell \quad \Rightarrow \quad |s^r_\pi(t+s) - s^r_\pi(t)| \le s,\ \forall t,s. \tag{4.23}$$

4. From (4.21) and (4.22), we obtain that for all i,j,

$$|Q_{ij}(m+\ell) - Q_{ij}(m)| \le \ell,\ \forall m,\ell \quad \Rightarrow \quad |q^r_{ij}(t+s) - q^r_{ij}(t)| \le s,\ \forall t,s. \tag{4.24}$$

From (4.21)-(4.24), we obtain that $\mathcal{X}(\cdot), x^r(\cdot)$ are 4-tuple of Lipschitz continuous functions on $[0,T]$. Thus, the probability measure $\mu^r(\cdot)$ is on the space of Lipschitz functions. Now, we

claim the following Lemma.

Lemma 3. *For any sequence $r_k \uparrow \infty$, the sequence of measures μ^{r_k} is tight.*

Proof. The support set of measure μ^r, for any $r \in \mathbb{R}_+$, is a 4-tuple of functions which are (i) Lipschitz continuous and hence equicontinous, (ii) uniformly bounded and (iii) with compact domain $[0,T]$ and image contained in $\mathbb{R}_+^d$, for some finite integer d. Hence, Arzela-Ascolli's Theorem implies that the closure of the support is compact on the space of functions (with domain $[0,T]$ and range $\mathbb{R}_+^d$) endowed with topology induced by metric of sup-norm. This in turn implies that for any sequence $r_k \uparrow \infty$, the sequence of measures μ^{r_k} is tight. This completes the proof of Lemma 3. □

From Lemma 3 and Prohorov's Theorem, we obtain that for any sequence of tight measures μ^{r_k} on a metric space, there exists a convergent subsequence $\mu^{r_{k_j}}$ such that $\mu^{r_{k_j}} \to \mu$; here μ is a probability measure with the same support. Next we study the measure μ. In particular, we show that μ concentrates all of its mass on solution of fluid model equations. As before, we divide the treatment in algorithm independent and algorithm dependent parts.

Algorithm-independent fluid model

We will show that

$$\mu\left(\ \{x(\cdot) \text{ satisfies equations (4.12)-(4.14)}\}\ \right) \ = \ 1.$$

Recall that inter-arrival times are IID (Assumption 1) and hence as shown in (4.7) with probability 1,

$$\lim_{r\to\infty} a^r(t) \quad = \quad \lambda t. \tag{4.25}$$

Since $\mu^{r_{k_j}} \Rightarrow \mu$,

$$\mu(a(t) = \lambda t) = 1.$$

From (4.25) and switch dynamics implies, with probability 1,

$$\lim_{r\to\infty} q^r(t) + d^r(t) \quad = \quad \lambda t, \tag{4.26}$$

Hence, $q(t), d(t)$ satisfy (4.26) with probability 1, under μ. This in turn implies that, $x(\cdot)$ satisfies (4.12) with probability 1 under μ. The equation (4.13) is satisfied trivially since all algorithms under consideration are non-idling. The only remaining equation is (4.14).

Fix i, j. We need to show that, under μ, with probability 1, when $q_{ij}(t, \omega) > 0$,

$$\dot{d}_{ij}(t) = \sum_{\pi \in \mathbb{P}} \pi_{ij} \dot{s}_\pi(t).$$

By continuity of $q_{ij}(\cdot)$, there exists a $\delta > 0$, such that $a = \min_{t' \in [t, t+\delta]} q_{ij}(t') > 0$ if $q_{ij}(t) > 0$. Then for any large enough j, we have

$$q_{ij}^{r_{k_j}}(t') \geq a/2 \quad \text{for} \quad t' \in [t, t+\delta] \quad \text{and} \quad r_{k_j} a/2 > 1.$$

Thus,

$$q_{ij}(r_{k_j} t') > 1 \quad \text{for} \quad t' \in [r_{k_j} t, r_{k_j}(t+\delta)].$$

Thus, for interval $[r_{k_j} t, r_{k_j}(t+\delta)]$, the queue $q_{ij}(\cdot)$ is non-empty. Hence in this interval, departure from $q_{ij}(\cdot)$ matches the amount of service it was given, that is,

$$d_{ij}(r_{k_j} t + 1) - d_{ij}(r_{k_j} t) \quad = \quad \sum_{\pi \in \mathbb{P}} \pi_{ij} \left(s_\pi(r_{k_j} t + 1) - s_\pi(r_{k_j} t) \right). \tag{4.27}$$

By definition, $r_{k_j} d_{ij}^{r_{k_j}}(\cdot) = d_{ij}(r_{k_j} \cdot)$. Hence, from (4.27) we obtain,

$$\frac{d_{ij}^{r_{k_j}}(t + r_{k_j}) - d_{ij}^{r_{k_j}}(t)}{1/r_{k_j}} \quad = \quad \sum_{\pi \in \mathbb{P}} \pi_{ij} \left(\frac{s_\pi(t + 1/r_{k_j}) - s_\pi(t)}{1/r_{k_j}} \right). \tag{4.28}$$

Now by letting $j \to \infty$, we obtain

$$\dot{d}_{ij}(t) = \sum_{\pi \in \mathbb{P}} \pi_{ij} \dot{s}_\pi(t).$$

Thus, $x(\cdot)$ satisfies (4.14) with probability 1 under μ. Hence, we have shown that under μ, (4.12)-(4.14) are satisfied with probability 1.

Algorithm-dependent fluid model

Next, we justify algorithm dependent fluid equations (4.19) for MWMf algorithm. For MWMf algorithm, the switch obeys the following equations.

$$S_\pi(m+1) = S_\pi(m) \quad \text{if} \quad \pi \cdot f(Q(m)) < \max_{\rho \in \mathbb{P}} \rho \cdot f(Q(m)), \quad m \in \mathcal{Z}_+. \tag{4.29}$$

Consider any r and t. By definition, $q^r(t) = Q(rt)/r$. Hence, from the Condition 1 we obtain

$$\pi \cdot f(Q(rt)) < \max_{\rho \in \mathbb{P}} \rho \cdot f(Q(rt)) \quad \Rightarrow \quad \pi \cdot f(q^r(t)) < \max_{\rho \in \mathbb{P}} \rho \cdot f(q^r(t)). \tag{4.30}$$

Now consider a time t and the sequence $\{r_{k_j}\}$. From (4.29) and (4.30), we obtain

$$s^{r_{k_j}}(t + 1/r_{k_j}) = s^{r_{k_j}}(t) \quad \text{if} \quad \pi \cdot f(q^{r_{k_j}}) < \max_{\rho \in \mathbb{P}} \rho \cdot f(q^{r_{k_j}}), \quad t \in \mathbb{R}_+. \tag{4.31}$$

Diving both sides of equation on the left in (4.31) by r_{k_j} and letting $j \to \infty$, gives the desired equation as follows.

$$\dot{s}_\pi = 0 \quad \text{if} \quad \pi \cdot f(q) < \max_{\rho \in \mathbb{P}} \rho \cdot f(q).$$

We conclude the following result.

Lemma 4. *Given $\epsilon > 0$ and T, for r large enough there exists a solution of fluid model equations, $x(\cdot)$, such that*

$$\Pr(|x^r(\cdot) - x(\cdot)|_T > \epsilon) < \epsilon.$$

4.2.3 Stability of MWMf

Now, we will use fluid model to prove stability of MWMf algorithm. In order to use fluid model equations to prove stability, we first define notion of weak stability of fluid model.

Definition 5 (Weak Stability). *The fluid model of a switch operating under a scheduling algorithm is said to be weakly stable if for every solution of fluid model equations $x(\cdot)$ is such that $q(t) = [0]$ for all $t \geq 0$, whenever $q(0) = [0]$.*

The following theorem relates notion of weak stability with the rate-stability of algorithm.

Theorem 11. *A switch operating under scheduling algorithm is rate-stable if the corresponding fluid model is weakly stable.*

Proof. Assume that fluid model is weakly stable. Hence, given $q(0) = [0]$, $q(t) = [0]$ for all $t > 0$. This means that for all the solutions of fluid model equations, we have corresponding $d(t) = \lambda t$ from equation (4.12) for all $t > 0$. Thus, there is a unique solution to fluid model equation, given by $(q(t), d(t), a(t)) = ([0], \lambda t, \lambda t)$ for all $t > 0$.

Now consider the case when $t = 1$, that is, $d(1) = \lambda$. Now by Lemma 4, uniqueness of fluid model solutions and recalling the definition of fluid scaling, we obtain that,

$$\begin{aligned} \lim_{r\to\infty} \frac{D(r)}{r} &= d(1) \\ &= \lambda, \end{aligned} \tag{4.32}$$

with probability 1. Restricting r to integers, we obtain that the departure process also has rate λ over discrete time, that is, the switch is rate-stable. This completes the proof of Theorem 11. □

Now we use the Theorem 11 to prove stability of MWMf. For this, we need to show that fluid model is weakly stable. For this purpose, we first define the following Lyapunov function:

$$L(q) = F(q) \cdot [1] = \sum_{i,j} F(q_{ij}) \quad \text{where} \quad F(x) = \int_{y=0}^{x} f(y)\, dy. \tag{4.33}$$

Though, L depends on the function f, we do not explicitly mention f in its notation. The definition of L will be clear from the context. Now, consider the following Lemma which shows that L is indeed a Lyapunov function of MWMf algorithm.

Lemma 5. *For a switch operating under the MWMf algorithm with admissible arrival rate-matrix λ, for any fluid model solution $q(t) \neq [0]$, $\frac{d}{dt} L(q(t)) < 0$.*

Proof. Recall that $q(t)$ is absolutely continuous, and note that $L(\cdot)$ is continuous; thus the derivative exists wherever the derivative $d/dt\, q(t)$ exists, which is almost everywhere. At such t, let

$$w^*(f(q(t)) = \max_{\pi\in\mathbb{P}} \pi \cdot f(q(t)).$$

Now consider the following.

$$\begin{aligned}
\frac{d}{dt}L(q(t)) &= f(q(t)) \cdot (\lambda - \sigma)^{+[q(t)=0]} \\
&= f(q(t)) \cdot (\lambda - \sigma(t)) \quad \text{since } f(z) = 0 \text{ whenever } z = 0 \qquad (4.34) \\
&\overset{(a)}{\leq} f(q(t)) \cdot \left(\lambda^* \left[\sum_{k=1}^{n^2} \alpha_k \pi_k \right] - \sigma(t) \right) \\
&= \lambda^* \left(\sum_{k=1}^{n^2} \alpha_k f(q(t)) \cdot \pi_k \right) - w^*(f(q(t))) \quad \text{from equation (4.19)} \qquad (4.35) \\
&\leq -(1 - \lambda^*) w^*(f(q(t))). \qquad (4.36)
\end{aligned}$$

The (a) follows from the fact that admissible doubly stochastic rate-matrix λ can be bounded component-wise as $\lambda \leq \lambda^* \sum_{k=1}^{n^2} \alpha_k \pi_k$, where $\alpha_k \in \mathbb{R}_+$, $\sum_k \alpha_k = 1$. Now since $q(t) \neq [0]$, $w^*(f(q(t))) > 0$. For admissible λ, $\lambda^* < 1$. Hence from (4.36) we obtain $\frac{d}{dt}L(q(t)) < 0$. This completes the proof of Lemma 5. □

Next, consider the following Lemma which will be useful to prove weak stability of fluid model equations under MWMf.

Lemma 6. *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be an absolutely continuous function with $f(0) = 0$. Assume that $\frac{d}{dt}f(t) \leq 0$ for almost every t (w.r.t. Lebesgue measure) such that $f(t) > 0$ and f is differentiable at t. Then, $f(t) = 0$ for almost every $t \geq 0$.*

Proof. For almost every $t \geq 0$, $f^2(t) - f^2(0) = 2\int_0^t f(s)\frac{d}{ds}f(s)ds \leq 0$, since $f(s)\frac{d}{ds}f(s) \leq 0$ a.e. in $[0, t]$. Now $f(0) = 0$ and $f(t) \geq 0$ imply that $f(t) = 0$ for almost every t. □

Now we state the result about rate-stability of MWMf algorithm.

Theorem 12. *Under any arrival process satisfying Assumption 1 with admissible rate-matrix λ, the switch operating under MWMf algorithm is rate-stable.*

Proof. From Theorem 11, it is sufficient to prove weak stability of fluid model equations in order to prove rate-stability of the switch operating under MWMf. Lemmas 5 and 6 imply that $L(q(t)) = 0$ for almost every t if $q(0) = 0$. By definition of $L(\cdot)$, this immediately implies that $q(t) = 0$ for almost every t when $q(0) = 0$. That is, the switch is weakly stable under MWMf algorithm. This completes the proof of Theorem 12. □

4.3 Equilibrium Analysis of Fluid Model

In the previous section, we obtained fluid model corresponding to a switch operating under MWMf algorithm. We used the fluid model solutions to prove rate stability of MWMf algorithms under arrival process when rate-matrix is admissible. Now, we study the fluid model solutions when some of the ports are critically loaded, that is,

$$|\{i \ : \ \lambda_{i\cdot} = 1\} \cup \{j \ : \ \lambda_{\cdot j} = 1\}| \geq 1.$$

In particular, we are interested in characterizing *invariant state* of fluid model equations, formally defined as follows.

Definition 6 (Invariant State.). *Consider a switch operating under an algorithm $\mathcal{A}$ with arrival rate matrix λ. We call a state $p \in \mathbb{M}_+$ as an invariant if the following holds for all fluid model solutions of such a switch:*

$$q(t) = p \ \Rightarrow \quad q(s) = p, \quad \forall s \geq t.$$

Recall that when λ is admissible, i.e. $\lambda^* < 1$, the invariant state is $q(t) = [0]$ as shown in Theorem 12. In this section, when one or more ports of switch are critically loaded, we seek to obtain (i) characterization of invariant states of fluid model equations of a switch, and (ii) time taken for the convergence to invariant states, starting from any initial state. In this section, we obtain answers to both (i) and (ii) under MWMf algorithm. We find that a state q is an invariant state if and only if it is the solution to a certain optimization problem whose the input data the set of workloads $q_{i\cdot}$ and $q_{\cdot j}$ of the initial state.

4.3.1 Preliminary Results about Matchings

Recall that the Birkhoff-von Neumann theorem says that the set of doubly stochastic matrices $\mathbb{S}$ forms a convex set, with the permutation matrices $\mathbb{P}$ as extreme points. Thus any doubly stochastic matrix a can be written as

$$a = \sum_{\pi \in \mathbb{P}} \gamma_\pi \pi \quad \text{where each} \quad \gamma_\pi \geq 0 \quad \text{and} \quad \sum_{\pi} \gamma_\pi = 1.$$

Furthermore, if the entries of a are all rational, then the γ_π may be chosen to be rational.

Many of our results concern maximum weight matchings. Given a weight matrix $q \in \mathbb{M}^*$, let $\mathcal{M}(q)$ be the set of maximum weight matchings, and let $M(q)$ be the matrix which indicates which entries are involved in a maximum weight matching:

$$M(q)_{ij} = \begin{cases} 1 & \text{if } \pi_{ij} = 1 \text{ for some } \pi \in \mathcal{M}(q) \\ 0 & \text{otherwise} \end{cases}$$

The set $\mathcal{M}(q)$ exhibits an important closure property:

Lemma 7. *Let $\pi \in \mathbb{P}$, and suppose $M(q)_{ij} = 1$ whenever $\pi_{ij} = 1$. Then $\pi \in \mathcal{M}(q)$.*

Proof. Define the matrix a by

$$a = \sum_{\rho \in \mathcal{M}(q)} \rho.$$

It is easy to see that $a - \pi$ has non-negative entries, and that its row and column sums are all equal, so by the Birkhoff-von Neuman decomposition

$$a = \pi + \sum_{\rho \in \mathbb{P}} \gamma_\rho \rho$$

where each $\gamma_\rho \geq 0$ and $\sum \gamma_\rho = |\mathcal{M}(q) - 1|$.

Let m be the weight of a maximum weight matching. By construction of a, $q \cdot a = |\mathcal{M}(q)|m$. On the other hand, by maximality, it must be that $q \cdot \pi \leq m$ and $q \cdot (a - \pi) \leq |\mathcal{M}(q) - 1|m$. If either of these inequalities are strict we get that $|\mathcal{M}(q)|m < |\mathcal{M}(q)|m$, a contradiction. Hence $q \cdot \pi = m$, and so $\pi \in \mathcal{M}(q)$. □

Let λ be doubly sub-stochastic. It can be augmented to form a doubly stochastic matrix $\lambda + \alpha$, where the matrix α satisfies

$$\alpha_{ij} > 0 \quad \text{if} \quad \lambda_{i\cdot} < 1 \text{ and } \quad \lambda_{\cdot j} < 1.$$

We will call such an α the matrix of arrival rates that is *complementary to* λ. (One way to obtain such an α is to start with $\alpha_{ij} = \varepsilon$ for the entries specified above, where $\varepsilon = n^{-1} \min_i(1 - \lambda_{i\cdot}) \wedge \min_j(1 - \lambda_{\cdot j})$, and then to add the 'deficit' amount of load according to the transport algorithm.)

*Here q denotes any positive weight matrix and not necessarily the queue-size matrix.

Lemma 8. *For given λ, let $q, r \in \mathbb{M}$ be such that $q_{ij} = r_{ij} = 0$ whenever $\lambda_{ij} = 0$. Let,*

$$r_{i\cdot} \geq q_{i\cdot} \quad \textit{if} \quad \lambda_{i\cdot} = 1 \qquad \textit{and} \qquad r_{\cdot j} \geq q_{\cdot j} \quad \textit{if} \quad \lambda_{\cdot j} = 1, \quad \forall\, i, j.$$

Then there exists a doubly stochastic matrix $\sigma \in \mathbb{S}$, a positive matrix $\varepsilon \in \mathbb{M}_+$, and a duration $t > 0$ such that

$$r = q + t(\lambda - \sigma) + \varepsilon. \tag{4.37}$$

Suppose that in addition

$$r_{i\cdot} \geq q_{i\cdot} \quad \textit{and} \quad r_{\cdot j} \geq q_{\cdot j} \quad \forall \;\; i, j.$$

Then for any augmentation λ^+ of λ, there exist σ, t and ε as above such that

$$r = q + t(\lambda^+ - \sigma) + \varepsilon. \tag{4.38}$$

Proof. We will first prove (4.37). The proof of (4.38) is very similar.

Let $\rho = \lambda - \delta(r - q)$ for sufficiently small $\delta > 0$. We will show that ρ is doubly sub-stochastic matrix with non-negative entries.

First, we'll show that all entries of ρ are non-negative, that is, $\rho_{ij} \geq 0$. Now, if $\lambda_{ij} > 0$, then by choosing δ small enough, ρ_{ij} can be made positive; else if $\lambda_{ij} = 0$ then trivially by constraints on q, r, we obtain $\rho_{ij} = 0$. Thus, $\rho \in \mathbb{M}_+$.

Next, we show that ρ is doubly stochastic, that is, $\rho_{i\cdot} \leq 1, \quad \rho_{\cdot j} \leq 1, \quad \forall i, j.$. Consider $\rho_{i\cdot}$.: either $\lambda_{i\cdot} < 1$, in which case $\rho_{i\cdot} < 1$ for sufficiently small δ; or $\lambda_{i\cdot} = 1$, in which case $r_{i\cdot} \geq q_{i\cdot}$ and $\rho_{i\cdot} \leq 1$. Similarly, $\rho_{\cdot j} \leq 1, \; \forall\; j$.

Now, ρ is doubly sub-stochastic non-negative matrix. Hence, there exists an augmentation of ρ, i.e. there exists a doubly stochastic matrix σ for which $\rho \leq \sigma$ component-wise. Then

$$q + \delta^{-1}(\lambda - \sigma) \leq q + \delta^{-1}(\lambda - \rho) = r.$$

This proves (4.37).

The proof of (4.38) is similar, with $\rho = \lambda^+ - \delta(r - q)$. It makes use of the fact that $\lambda^+_{ij} = 0$ implies $\lambda_{ij} = 0$. □

4.3.2 The Basic Maximum Weight Matching

We will start with some analysis of the equilibrium states of the basic Maximum Weight Matching algorithm, MWM-1. We will only give partial results, because this algorithm is fully described in section 4.3.3 as a special case of the generalized MWMf algorithm. Nonetheless, it is useful to build up some intuition by working with a special case—we have found that the results described in this section are intuitively appealing, even though they are in some sense restrictive. Additionally, the results of this section highlight some of the interesting combinatorial characterization of invariant state of the MWM-1 algorithm. In the rest of the paper, whenever we use MWM, we mean MWM-1 unless specified.

Consider a single server serving many queues. The server decides which queue to serve every time. Under a work conserving policy when the arrival rate is no more than the service rate, the net work does not increase. Further, under the longest queue first policy, the size of the longest queue does not increase when arrivals and services are deterministic (i.e. on fluid scale). Based on this, in the context of a switch, one would expect the weight of maximum weight matching to be non-increasing under admissible deterministic arrivals. Next, we present an example (prompted by discussions with Frank Kelly and Mark Walters) that contradicts this expectation. This should caution us against making any strong claims about optimality of MWM algorithm.

Example 6. *Let the matrix of arrival rates be*

$$\lambda = \frac{1}{5}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & & & \\ 1 & & 4 & & \\ 1 & & & 4 & \\ 1 & & & & 4 \end{pmatrix}$$

(where blank entries are to be read as 0). Suppose that at some point in time the system reaches the state

$$q = \begin{pmatrix} 6 & 11 & 11 & 11 & 11 \\ 11 & 9 & & & \\ 11 & & 9 & & \\ 11 & & & 9 & \\ 11 & & & & 9 \end{pmatrix}$$

The weight of the maximum-weight matching here is $m(q) = 49$. *There are four matchings which have this weight. One can show that the MWM algorithm will serve these four, in equal proportion in this example, giving service matrix*

$$\sigma = \frac{1}{4}\begin{pmatrix} & 1 & 1 & 1 & 1 \\ 1 & 3 & & & \\ 1 & & 3 & & \\ 1 & & & 3 & \\ 1 & & & & 3 \end{pmatrix} \quad \textit{and} \quad \lambda - \sigma = \frac{1}{20}\begin{pmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 1 & & & \\ -1 & & 1 & & \\ -1 & & & 1 & \\ -1 & & & & 1 \end{pmatrix}$$

which will be applied until the system reaches the state

$$r = \begin{pmatrix} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & & & \\ 10 & & 10 & & \\ 10 & & & 10 & \\ 10 & & & & 10 \end{pmatrix}$$

The maximum-weight matching here is $m(r) = 50$. *In other words, under the operation of MWM, the weight of the maximum-weight matching has* increased.

One can also show that, once the system has reached state r, *it will remain in that state thereafter.*

The key idea in this section is of invariant states. Consider a switch with doubly sub-stochastic arrival rates λ.

Theorem 13. *Suppose* $\lambda > 0$ *in each component. If* q *is an invariant state then it is the unique solution to the linear program MWM*$^+$*-LP(*q*), which is to*

$$\begin{aligned} &\textit{minimize} \quad m(r) = \max_{\pi \in \mathbb{P}} \pi \cdot r \quad \textit{over } r \in \mathbb{M}_+ \\ &\textit{such that} \quad r_{i\cdot} \geq q_{i\cdot} \quad \textit{if } \lambda_{i\cdot} = 1, \textit{ and} \\ &\qquad\qquad r_{\cdot j} \geq q_{\cdot j} \quad \textit{if } \lambda_{\cdot j} = 1 \end{aligned}$$

Conversely, if q *solves MWM*$^+$*-LP(*q*) then it is an invariant state.*

(This restriction on λ is necessary. Example 6 is a case where some component of λ is equal to zero, and the system does *not* minimize the maximum weight matching. In cases

like this, the results that follow are not appropriate. Section 4.3.3 deals with this in a more sophisticated way.)

The intuition behind the result is as follows. The MWM algorithm only offers service to matchings which have maximum weight. If this includes only some of the queues, then those queues have more service than arrivals, so they decrease, which pulls down $m(q)$. This explains the objective function. If some row i has $\lambda_{i\cdot} = 1$ then the total service rate for that row is equal to the total arrival rate, so $q_{i\cdot}$, the workload at input i, can never decrease, it can only increase (but, subject to this, the workload can be rearranged among the queues in row i). Otherwise the total service rate is greater than the total arrival rate, which means that the workload can also decrease. This explains the constraints.

In proving the theorem, it is helpful to use a more explicit characterization of invariant states.

Definition 7 (MWM$^+$-endstate.). *We say that q is an* MWM$^+$-endstate *if*

1. *Every queue is involved in some maximum weight matching, i.e. $M(q)_{ij} = 1$ for all i, j, i.e. $\mathcal{M}(q) = \mathbb{P}$.*

2. *If $\lambda_{i\cdot} < 1$ and $\lambda_{\cdot j} < 1$ then $q_{ij} = 0$.*

Lemma 9. *If $\lambda > 0$, then q is an invariant state if and only if q is an MWM$^+$-endstate.*

Proof. The Theorem 15 relates invariant state and endstate in the context of general MWMf algorithm. Taking $f(x) = x$, and restricting to the case of $\lambda > 0$, Theorem 15 immediately implies the Lemma. □

Now we are ready to prove the Theorem 13.

Proof of Theorem 13. Lemma 10 shows that if q solves MWM$^+$-LP(r) then q satisfies both requirements of an endstate. Hence Lemma 9 implies the converse of Theorem 13, i.e. if q is a solution of MWM$^+$-LP(r) for some r, then q is an invariant state. Next, we proceed to prove that if q is an invariant state then it satisfies MWM$^+$-LP(q).

Now suppose q is an endstate and consider MWM$^+$-LP(q). First, a solution exists, since the objective is a continuous function, and we can take the domain to be contained in the bounded set $\{r : r_{ij} \leq m(q)\}$.

Now let r be any solution to MWM$^+$-LP(q), $r \neq q$. By Lemma 11 given below, $r_{i\cdot} \geq q_{i\cdot}$ and $r_{\cdot j} \geq q_{\cdot j}$ for all i and j. Hence by Lemma 8 we can write

$$r = q + t(\lambda^+ - \sigma) + \varepsilon$$

for some doubly stochastic σ, where $t > 0$ and either $\lambda^+ \neq \sigma$ or $\varepsilon > 0$ in some component.

If $\lambda^+ \neq \sigma$, then by Lemma 12 there is some matching π with $\pi \cdot (\lambda^+ - \sigma) > 0$, which implies that $\pi \cdot r > \pi \cdot q$. By assumption, q is an endstate, and so all matchings (and matching π in particular) have maximum weight for q, thus $m(r) > m(q)$, which contradicts the optimality of r. Otherwise $\lambda^+ = \sigma$ and $\varepsilon > 0$ in some component, in which case it is easy to see that $m(r) > m(q)$, the same contradiction.

Thus $r = q$. Therefore q is the unique solution to MWM$^+$-LP(q). □

Lemma 10. *Suppose q solves MWM$^+$-LP(r) for some r. Then all matchings are maximum weight matchings for q; furthermore, if $\lambda_{i\cdot} < 1$ and $\lambda_{\cdot j} < 1$ then $q_{ij} = 0$.*

Proof. Suppose that not all matchings are maximum weight matchings for q. By Lemma 7, there is some queue which is not part of any maximum weight matching. Without loss of generality, suppose it is q_{11}, i.e. suppose $M(q)_{11} = 0$. Define the matrix

$$\delta = \left(\begin{array}{c|ccc} \varepsilon(n-1) & -\varepsilon & \cdots & -\varepsilon \\ \hline -\varepsilon & & & \\ \vdots & & \varepsilon(n-1)^{-1} & \\ -\varepsilon & & & \end{array}\right)$$

and let

$$q' = (q + \delta)^+$$

Since all the row and column sums of δ are equal to 0, q' is feasible for MWM$^+$-LP(r). We will now argue that $m(q')$ is strictly less than $m(q)$, contradicting the optimality of q.

Consider any π which is a maximum weight matching for q. Suppose that $\pi_{i1} = \pi_{1j} = 1$, i.e. that π matches input port i to output port 1, and input port 1 to output port j. By assumption, q_{11} is not part of any maximum weight matching, so $i \neq 1$ and $j \neq 1$. Therefore π must comprise q_{i1} from the first column, q_{1j} from the first row, and $n-2$ queues from the remaining rows and columns. We will argue below that at least one of q_{i1} and q_{1j} is strictly

positive. If this is so, (and if ε is sufficiently small), we find that

$$\begin{aligned}\pi \cdot q' &\leq \pi \cdot q + (n-2)\frac{\varepsilon}{n-1} - \varepsilon \\ &< \pi \cdot q.\end{aligned}$$

(The inequality comes from the fact that one of q_{i1} and q_{1j} may be 0.) Therefore $m(q') < m(q)$.

It remains to show that at least one of q_{i1} and q_{1j} is strictly positive. Suppose not. Consider the matching ρ which is like π except that $\rho_{11} = \rho_{ij} = 1$ and $\rho_{i1} = \rho_{1j} = 0$. In words, ρ is like π except that it maps input 1 to output 1, and input i to output j. The weight of matching ρ is

$$\rho \cdot q = \pi \cdot q - \big(q_{i1} + q_{1j}\big) + \big(q_{11} + q_{ij}\big) \geq \pi \cdot q.$$

(The inequality comes from the fact that the two queues q_{i1} and q_{1j} are both 0.) Therefore q_{11} is part of a maximum weight matching, which contradicts our premise. Thus it cannot be that both the queues are 0.

Now, we will show that if $\lambda_{i\cdot} < 1$ and $\lambda_{\cdot j} < 1$ then $q_{ij} = 0$. Suppose not. Let q' be like q, but with $q'_{ij} = 0$. Then q' is feasible for MWM$^+$-LP(r), and since no queues have increased, $m(q') \leq m(q)$. If $m(q') < m(q)$ then q is not optimal, a contradiction. If $m(q') = m(q)$, then q' also solves MWM$^+$-LP(r), so by the above all matchings in q' have maximal weight. And yet there is some matching π (any matching involving q'_{ij} will do) for which $\pi \cdot q' < m(q')$, a contradiction. Thus $q_{ij} = 0$. □

Note that we can choose the matrix δ, which leads to a decrease in the maximum weight, as a function of only q. Now, if we knew λ, we could choose a service matrix σ such that $\lambda - \sigma \leq \delta$, and hence achieve a strict decrease in $m(q)$ (unless q is already an endstate). This means that $m(q)$ is a Lyapunov function for some scheduling algorithm (though probably not for any online scheduling algorithm).

Lemma 11. *Let q be an endstate, and let r solve MWM$^+$-LP(q). Then $r_{i\cdot} \geq q_{i\cdot}$ and $r_{\cdot j} \geq q_{\cdot j}$ for all i and j (not just for the critically loaded ports).*

Proof. Without loss of generality, assume there are critical ports. By assumption q is an endstate, and by Lemma 10 r is too. The two requirements of an endstate imply that any

endstate s must be of the form

$$s = \left(\begin{array}{ccc|ccc} x_1+y_1 & \cdots & x_C+y_1 & y_1 & y_1 & \cdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \\ x_1+y_R & \cdots & x_C+y_R & y_R & y_R & \cdots \\ \hline x_1 & \cdots & x_C & & & \\ x_1 & \cdots & x_C & & 0 & \\ \vdots & & \vdots & & & \end{array}\right)$$

We have arranged the rows and columns so that the first R input ports and the first C output ports are critical. To see that it must have this form, consider a permutation π for which $\pi_{ij} = \pi_{kl} = 1$ where input port i and output port j are both subcritical, and consider also the permutation ρ which is like π but with $\rho_{il} = \rho_{kj} = 1$ instead of $\pi_{ij} = \pi_{kl} = 1$. Since s is an endstate, $s_{ij} = 0$, and both ρ and π are maximum weight matchings, hence

$$s_{ij} + s_{kl} = s_{kl} = s_{il} + s_{kj}.$$

By considering various possibilities for k and l we arrive at the above general form for s.

Let the terms in this representative matrix be $x_1, \ldots$ and $y_1, \ldots$ for r, and $u_1, \ldots$ and $v_1, \ldots$ for q. Let $x = x_1 + \cdots + x_C$ etc.

Suppose the result of the lemma is false. Then (without loss of generality) $u > x$. What does this mean for y?

First, write down the equations which come from the fact that r is feasible for MWM$^+$-LP(q):

$$(n-R)x + (Rx + Cy) \geq (n-R)u + (Ru + Cv)$$
$$(n-C)y + (Rx + Cy) \geq (n-C)v + (Ru + Cv)$$

Rearranging, we obtain

$$Rx + Cy > Ru + Cv$$
$$Rx + ny \geq Ru + nv \quad \Longrightarrow \quad y > v$$

and hence

$$(n-C)y + (Rx + Cy) > (n-C)v + (Ru + Cv)$$

i.e.

$$\sum_{i=1}^{R} x + ny_i > \sum_{i=1}^{R} u + nv_i.$$

Thus there is some $i \leq R$ for which

$$x + ny_i > u + nv_i \tag{4.39}$$

and in particular

$$y_i > n^{-1}(u - x) + v_i \geq 0. \tag{4.40}$$

Consider a new state r' which is like r except that

$$y_i' = y_i - \varepsilon$$

for $\varepsilon > 0$ sufficiently small. By (4.39) and (4.40), r' is feasible for MWM$^+$-LP(q). Certainly $m(r') \leq m(r)$, since $\varepsilon > 0$; but by optimality of r, $m(r') \geq m(r)$; hence $m(r') = m(r)$, so r' is optimal. Since r' is optimal, by Lemma 10 all matchings in r' have maximum weight. However there is some matching π (any matching for which $\pi_{i\,(C+1)} = 1$ will do) which had maximum weight for r, but for which

$$\pi \cdot r' < \pi \cdot r = m(r) = m(r').$$

This contradicts the fact that all matchings in r' have maximum weight. So it cannot be that $u > x$.

Applying the same argument to columns, it cannot be that $v > y$. This completes the proof. □

Lemma 12. *Let $\varepsilon \in \mathbb{S}(0)$. If ε is not identically zero, there exists some permutation π such that $\pi \cdot \varepsilon > 0$.*

Proof. Suppose not. Then $\pi \cdot \varepsilon \leq 0$ for all π.

Suppose that $\pi \cdot \varepsilon = 0$ for all π. Since $\varepsilon \in \mathbb{S}(0)$, it has a Birkhoff-von Neuman decomposition

$$\varepsilon = \sum_{\pi} \gamma_\pi \pi \quad \text{where} \quad \sum_{\pi} \gamma_\pi = 0.$$

Therefore

$$\varepsilon \cdot \varepsilon = \sum_{\pi} \gamma_\pi \pi \cdot \varepsilon = 0,$$

which contradicts the assumption that ε is not identically zero.

Therefore $\pi \cdot \varepsilon < 0$ for some π. Now consider the family of permutations π^k defined by $\pi^k(i) = \pi(i) + k \mod n$. Certainly $\pi = \pi^0$; and it is easy to see that the matrix $\tau = \pi^0 + \cdots + \pi^{n-1}$ is identically equal to 1. Thus $\tau \cdot \varepsilon = 0$. Since $\pi^0 \cdot \varepsilon < 0$ by assumption, it must be that $\pi^k \cdot \varepsilon > 0$ for some k. □

4.3.3 MWMf

In this section we will prove results about critically loaded fluid model solutions of the MWMf algorithm. We will exhibit a Lyapunov function for the system state, and we will characterize invariant states as the solution to an optimization problem whose objective is the Lyapunov function.

First recall some notation. The weight function f is strictly increasing real valued function with $f(0) = 0$, satisfying Condition 1. Let $q \in \mathbb{M}_+$ be a state of the system. Let $\mathcal{M}(f(q))$ be the set of maximum weight matches on $f(q)$, and let $M(f(q))$ be the matrix which indicates which queues are involved in some maximum weight matching. Let λ be the doubly sub-stochastic matrix of arrival rates. Let $m(f(q))$ be weight of the maximum weight matching,

$$m(f(q)) = \max_{\pi \in \mathbb{P}} \pi \cdot f(q).$$

Let α be a complementary arrival matrix, that is, a matrix $\alpha \in \mathbb{M}_+$ such that $\lambda^+ = \lambda + \alpha$ is doubly stochastic, and furthermore $\alpha_{ij} > 0$ whenever $\lambda_{i\cdot} \vee \lambda_{\cdot j} < 1$. It must be that $\alpha_{ij} = 0$ whenever $\lambda_{i\cdot} = 1$ or $\lambda_{\cdot j} = 1$. For MWMf algorithm we again use the following Lyapunov function.

$$L(q) = F(q) \cdot 1 = \sum_{i,j} F(q_{ij}) \quad \text{where} \quad F(x) = \int_{y=0}^{x} f(y)\, dy. \tag{4.41}$$

Though, L depends on the function f, we do not explicitly mention f in its notation. The

definition of L will be clear from the context. Define the convex optimization problem MWMf-CP(q) to be

$$\begin{array}{ll} \text{minimize} & L(r) \quad \text{over } r \in \mathbb{M}_+ \\ \text{such that} & r_{i\cdot} \geq q_{i\cdot} \text{ if } \lambda_{i\cdot} = 1 \\ & r_{\cdot j} \geq q_{\cdot j} \text{ if } \lambda_{\cdot j} = 1 \\ & r_{ij} = 0 \text{ if } \lambda_{ij} = 0 \end{array}$$

Note that the objective function is strictly convex as $f(\cdot)$ is strictly increasing function on $\mathbb{R}_+$. Define

$$\mathcal{B}(q) \stackrel{\triangle}{=} \{r \in \mathbb{M}_+ \ : \ r_{ij} \ \leq \ q_{\cdot\cdot}, \ \ \forall i, j\}.$$

It is easy to see that we can take the domain of r to be contained $\mathcal{B}(q)$ for the purpose of optimization. By definition, $\mathcal{B}(q)$ is bounded and hence the optimization problem has a unique solution. Thus, optimization problem MWMf-CP(q) can be seen as a map from $q \in \mathbb{M}_+$ to $\mathbb{M}_+$. This leads to the following definition of Lifting Map.

Definition 8 (Lifting Map). *The lifting map $\Delta^f : \mathbb{M}_+ \to \mathbb{M}_+$, maps q to the unique solution of optimization problem MWMf-CP(q), denoted by $\Delta^f(q)$.*

Next, we state the characterization of invariant state under MWMf algorithm and its relation to the Lyapunov function L.

Theorem 14. *For a switch operating under the MWMf algorithm,*

(a) For any fluid model solution $q(t)$, $\frac{d}{dt} L(q(t)) \leq 0$,

(b) q is an invariant state if and only if it solves MWMf-CP(q), and

(b) $q(t)$ is an invariant state if and only if $\frac{d}{dt} L(q(t)) = 0$.

Proof. We present the proof of the (a) first, followed by (b) and (c).

Proof of (a): $\mathbf{\frac{d}{dt} L(q(t)) \leq 0}$. Recall that $q(t)$ is absolutely continuous, and note that $L(\cdot)$ is continuous; thus the derivative exists wherever the derivative $d/dt\, q(t)$ exists, which is almost

everywhere. At such points,

$$\begin{aligned}
\frac{d}{dt}L(q(t)) &= f(q)\cdot(\lambda-\sigma)^{+[q=0]} \\
&= f(q)\cdot(\lambda-\sigma) \quad \text{since } f(z)=0 \text{ whenever } z=0 \\
&\le f(q)\cdot(\lambda^{+}-\sigma) \quad \text{since } \lambda\le\lambda^{+} \qquad (4.42)\\
&= f(q)\cdot\lambda^{+}-m(f(q)) \quad \text{by the fluid model equation (4.19)} \\
&= \sum_{\pi\in\mathbb{P}}\gamma_{\pi}f(q)\cdot\pi-m(f(q)) \quad \text{by decomposition of } \lambda^{+} \\
&\le m(f(q))-m(f(q)) \quad \text{since } m(f(q)) \text{ is maximum weight} \qquad (4.43)\\
&= 0.
\end{aligned}$$

Proof of (b): q invariant $\Leftrightarrow$ q solves MWMf-CP(q).
($\Leftarrow$) Suppose that q solves MWMf-CP(q). Let $q(t)$ be any fluid model solution with $q(0)=q$. Now $d/dt\,L(q(t))\le 0$ by (a). Now, we claim that $q(t)$ is a feasible solution to MWMf-CP(q) for all t. If this is so then $d/dt\,L(q(t))=0$ by optimality of q, and each $q(t)$ is also an optimal solution. But the optimum is unique. Hence $q(t)=q$ for all q, i.e. q is invariant.
We still need to verify that $q(t)$ is feasible for all t. According to the fluid equations,

$$\dot{q}(t)=\big(\lambda-\sigma(t)\big)^{+[q(t)=0]}$$

If $\lambda_{i\cdot}=1$ then

$$\dot{q}_{i\cdot}(t)\ge\lambda_{i\cdot}-\sigma_{i\cdot}(t)=0.$$

Thus, $q_{i\cdot}(t)\ge q_{i\cdot}(0)$; similarly for $q_{\cdot j}(t)$. Also, if $\lambda_{ij}=0$ then

$$\dot{q}_{ij}(t)\le 0$$

and so (as $q_{ij}=0$) $q_{ij}(t)=0$. Thus $q(t)$ is a feasible solution to MWMf-CP(q).

($\Rightarrow$) Now suppose that q is an invariant state. Let $q(0)=q$. Then, $d/dt\,L(q(t))=0$. Hence the (4.42) and (4.43) must be equalities, which implies that

$$f(q)\cdot\lambda=m(f(q)).$$

Now let r be any feasible solution to MWMf-CP(q) and suppose $r\neq q$. By Lemma 8, we can

write

$$r = r' + \varepsilon \quad \text{where} \quad r' = q + t(\lambda - \sigma).$$

where σ is doubly stochastic, $t > 0$ and $\varepsilon \geq 0$; and either $\lambda \neq \sigma$ or $\varepsilon > 0$ in some component. Consider the family of states

$$s(u) = q + u(\lambda - \sigma), \quad u \in [0, t]$$

giving $s(0) = q$ and $s(t) = r'$. It is the case that

$$\begin{aligned}
\frac{d}{du} L(s(u)) \Big|_{u=0} &= f(q) \cdot (\lambda - \sigma) \\
&= f(q) \cdot \lambda - f(q) \cdot \sigma \\
&= m(f(q)) - f(q) \cdot \sigma \quad \text{by (4.3.3)} \\
&\geq m(f(q)) - m(f(q)) \quad \text{by decomposing } \sigma \text{ into permutations} \\
&= 0.
\end{aligned}$$

Now, $L(q(u))$ is strictly convex as a function of u, so if $\lambda \neq \sigma$ then $L(r') = L(q(t)) > L(q(0)) = L(q)$, and since L is increasing, $L(r) = L(r' + \varepsilon) > L(q)$. Otherwise $\lambda = \sigma$ and $\varepsilon > 0$ in some component, so again $L(r) = L(r' + \varepsilon) > L(q)$. We have shown that if $r \neq q$ then $m(f(r)) > m(f(q))$, i.e. that q solves MWMf-CP(q).

Proof of (c): $q(t)$ is not invariant $\Leftrightarrow \frac{\mathbf{d}}{\mathbf{dt}} \mathbf{L}(\mathbf{q}(\mathbf{t})) < \mathbf{0}$.

($\Leftarrow$) This is equivalent to the statement that if $q(t)$ is invariant then $\frac{d}{dt} L(t) = 0$, which is true by definition of invariant state.

($\Rightarrow$) This is equivalent to the statement that $q \stackrel{\triangle}{=} q(t)$ is not invariant and the derivative is equal to zero. As we argued above, $f(q) \cdot \lambda = m(f(q))$ and hence q solves MWMf-CP(q). As we argued in (a), if q solves MWMf-CP(q) then it is an invariant state. This completes the proof of (c).

Thus, we have proved (a)-(c) as claimed above. □

Next we present an alternative characterization of invariant states. We define MWMf-endstate as follows.

Definition 9 (MWMf-endstate.). *A state q is an MWMf-endstate if*

1. $M(f(q))_{ij} = 1$ *if* $\lambda_{ij} > 0$,

2. $M(f(q))_{ij} = 1$ *if both* $\lambda_{i\cdot} < 1$ *and* $\lambda_{\cdot j} < 1$,

3. $q_{ij} = 0$ *if both* $\lambda_{i\cdot} < 1$ *and* $\lambda_{\cdot j} < 1$.

Note that, for $\lambda > 0$ in all components and $f(x) = x$, the MWMf-endstate is the same as MWM$^+$-endstate as defined in the previous section. Next, we state the result that relates MWMf-endstate and an invariant state of MWMf algorithm.

Theorem 15. *A state q is an MWMf-endstate if and only if it is an invariant state for MWMf algorithm.*

Proof. From Theorem 14, q is invariant if and only if for $q(t) = q$, $\frac{d}{dt} L(q(t)) = 0$. Hence from (4.42) and (4.43), q is invariant if and only if $f(q) \cdot \lambda = m(f(q))$. Hence it is sufficient to prove that q is an MWMf-endstate if and only if $f(q) \cdot \lambda = m(f(q))$.
$\mathbf{q}$ *is an MWMf-endstate* $\Rightarrow \mathbf{f(q)} \cdot \lambda = \mathbf{m(f(q))}$. First write

$$f(q) \cdot \lambda = f(q) \cdot \lambda^+ - f(q) \cdot \alpha,$$

where as before α is complementary matrix and λ^+ is doubly stochastic, meaning that it has a decomposition

$$\lambda^+ = \sum_{\pi \in \mathbb{P}} \gamma_\pi \pi \quad \text{where } \textstyle\sum \gamma_\pi = 1 \text{ and each } \gamma_\pi \geq 0.$$

By property (3) of MWMf-endstate and the property of α that $\alpha_{ij} > 0$ if and only if $\lambda_{i\cdot} < 1$ and $\lambda_{\cdot j} < 1$ yields $f(q) \cdot \alpha = 0$. Hence, we are only required to show $f(q) \cdot \lambda^+ = m(f(q))$ in order to prove that q is invariant.
Consider λ^+. If $\lambda^+_{ij} > 0$ then there are two possibilities:

1. $\lambda_{ij} = \lambda^+_{ij} > 0$ then by property (1) of an MWMf-endstate, $M(f(q))_{ij} = 1$.

2. $\lambda_{ij} < \lambda^+_{ij}$ then $\alpha_{ij} > 0$ in which case $M(f(q))_{ij} = 1$ by property (2) of MWMf-endstate.

Thus, $\lambda^+_{ij} > 0$ implies $M(f(q))_{ij} = 1$. Now in the decomposition of λ^+, if $\gamma_\pi > 0$ then $\lambda^+_{ij} > 0$ whenever $\pi_{ij} = 1$ and so by the above $M(q)_{ij} = 1$. Thus, if $\gamma_\pi > 0$ then $\pi \in \mathcal{M}(f(q))$ by Lemma 7, i.e. $f(q) \cdot \pi = m(f(q))$. Therefore

$$f(q) \cdot \lambda^+ = \sum_{\pi \in \mathbb{P}} \gamma_\pi f(q) \cdot \pi = m(f(q)).$$

Thus, if q is an MWMf-endstate then $f(q) \cdot \lambda = m(f(q))$.

q *is not an MWMf-endstate* $\Rightarrow \mathbf{f(q)} \cdot \lambda < \mathbf{m(f(q))}$. If q is not an MWMf-endstate then one of the three properties of MWMf-endstate must fail.

1. If property (1) of MWMf-endstate fails, then $M(f(q))_{ij} = 0$ and $\lambda_{ij} > 0$ for some i, j. Thus $\lambda^+_{ij} > 0$, and so in the decomposition of λ^+ there must be some $\pi \notin \mathcal{M}(f(q))$ with $\gamma_\pi > 0$. Since this π is not a maximum weight matching, $f(q) \cdot \pi < m(f(q))$, and so
$$f(q) \cdot \lambda \leq f(q) \cdot \lambda^+ < m(f(q)).$$

2. If property (2) of MWMf-endstate fails, then $M(f(q))_{ij} = 0$ and $\alpha_{ij} > 0$ for some i, j. Thus, $\lambda^+_{ij} > 0$ with the same consequences as above.

3. If property (3) of MWMf-endstate fails, then $q_{ij} > 0$ and $\alpha_{ij} > 0$ for some i, j. Thus $f(q) \cdot \alpha > 0$. Also $f(q) \cdot \lambda^+ \leq m(f(q))$, from the decomposition of λ^+ and the fact that $f(q) \cdot \pi \leq m(f(q))$ for all π. Hence
$$f(q) \cdot \lambda \leq m(f(q)) - f(q) \cdot \alpha < m(f(q)).$$

From above, if q is not an MWMf-endstate then $f(q) \cdot \lambda < m(f(q))$. □

The last result of this section concerns the speed of convergence. Its relevance will not become clear until we come to prove a heavy traffic limit theorem. First some notation. Define

$$\mathcal{D}(q) = \{r \in \mathbb{M}_+ \ : \ L(r) \leq L(q)\}.$$

Note that if $q(0) = q$, then by Theorem 14(a), $q(t) \in \mathcal{D}(q)$. Given $q(0) = q$, Theorem 14 implies that $q(t)$ converges to an invariant state. Let $\mathbf{1} \in \mathbb{M}_+$ denote the matrix with all entries 1. Then $\mathcal{D}(\mathbf{1})$ is a closed and bounded (and hence compact) set in $\mathbb{M}_+$. Consider the following definitions:

$$\mathcal{I} = \{q \in \mathcal{D}(\mathbf{1}) : \ \Delta^f(q) = q\},$$

and

$$\mathcal{I}(\delta) = \{q \in \mathcal{D}(\mathbf{1}) : \exists r \in \mathcal{I} \text{ s.t. } \|r - q\| < \delta\}.$$

Note that both $\mathcal{I}$ and $\mathcal{I}(\delta) \cap \mathcal{D}(\mathbf{1})$ are closed and bounded set. Further, $\mathcal{I}$ as well as $\mathcal{I}(\delta)$ are strictly contained inside $\mathcal{D}(\mathbf{1})$ for small enough δ. Now consider a function $g : \mathbb{M}_+ \to \mathbb{R}_+$

where

$$g(q) = m(f(q)) - f(q) \cdot \lambda.$$

Note that, given f and λ, g is a function on $\mathbb{M}_+$ and $g(q) = -\frac{d}{dt} L(q(t))$ for $q(t) = q$. Further, $g(q) > 0$ for $q \in \mathcal{I}(\delta)^c \cap \mathcal{D}(\mathbf{1})$ from Theorem 14. Now g is a function and hence it achieves infimum inside the closed and bounded set $\mathcal{I}(\delta)^c \cap \mathcal{D}(\mathbf{1})$, which is strictly positive. Let $\eta(\delta) > 0$ be this infimum of g. Finally, define

$$T(\epsilon) = \inf\{t \geq 0 : q(0) \in \mathcal{D}(\mathbf{1}); \quad q(t) - \Delta^f(q(t)) \leq \epsilon\}.$$

Now we state the following result.

Lemma 13. *For any given $\epsilon > 0$, there exists a $\delta(\epsilon) > 0$ such that*

$$T(\epsilon) \quad \leq \quad \frac{L(\mathbf{1})}{\eta(\delta(\epsilon))}. \tag{4.44}$$

Proof. Recall definition $\Delta^f(\cdot)$. The $\Delta^f(\cdot)$ is uniformly on a bounded set $\mathcal{D}(\mathbf{1})$. Hence, for any $\epsilon > 0$, there exists $0 < \delta(\epsilon) \leq \frac{\epsilon}{2}$ such that for $q_1, q_2 \in \mathcal{D}(\mathbf{1})$,

$$\|q_1 - q_2\| < \delta(\epsilon) \;\Rightarrow\; \|\Delta^f(q_1) - \Delta^f(q_2)\| < \frac{\epsilon}{2}.$$

Consider any $q_1 \in \mathcal{I}(\delta(\epsilon))$. From definition, there exists an $r \in \mathcal{I}$ such that

$$\|q_1 - r\| < \delta(\epsilon) \quad \Rightarrow \quad \|\Delta^f(q_1) - \Delta^f(r)\| < \epsilon/2. \tag{4.45}$$

But $\Delta^f(r) = r$ and $\delta(\epsilon) < \frac{\epsilon}{2}$ by definition. Hence,

$$\|q_1 - \Delta^f(q_1)\| \quad < \quad \epsilon. \tag{4.46}$$

The (4.46) implies that $T(\epsilon, q)$ is bounded above by the time it takes for $q(t)$ to reach $\mathcal{I}(\delta(\epsilon))$ given $q(0) = q$. Now if $q \in \mathcal{I}(\delta(\epsilon))$ then trivially (4.44) is satisfies. If $q \notin \mathcal{I}(\delta(\epsilon))$, then for all t such that $q(t) \notin \mathcal{I}(\delta(\epsilon))$,

$$\frac{d}{dt} L(q(t)) = -g(q(t)) < -\eta(\delta(\epsilon)).$$

Since $L(q(t)) \leq L(q) \leq L(\mathbf{1})$, we obtain the (4.45). □

4.4 Heavy Traffic and State Space Collapse

In this section state our result of State Space Collapse of IQ Switch operating under MWMf algorithm under Heavy Traffic regime. We first define the Heavy Traffic scaling and some required notations.

4.4.1 Heavy Traffic Scaling

Consider a sequence of IQ switch systems, indexed by $r \in \mathbb{R}_+$, satisfying Assumptions 1 and 2. The arrival rate matrix of r^{th} system, λ^r, is

$$\lambda^r \quad = \quad \lambda - \frac{1}{r}\Phi, \tag{4.47}$$

where $\Phi \in \mathbb{M}_+$ is a fixed constant matrix. The (4.47) suggests that,

$$\lim_{r\to\infty} \lambda^r = \lambda.$$

Additionally, λ is such that one or more of $2n$ ports (inputs and outputs) are critically loaded, i.e.

$$|\{i \ : \ \lambda_{i.} = 1\} \cup \{j \ : \ \lambda_{.j} = 1\}| \quad \neq \quad 0. \tag{4.48}$$

Let $\mathcal{X}^r(m),\ m \in \mathcal{Z}_+$ be the tuple describing dynamics of the r^{th} system. Under Heavy Traffic scaling, our interest is in studying the following scaled quantity.

$$\hat{x}^r(t) \quad = \quad \frac{\mathcal{X}^r(r^2 t)}{r}, \quad t \in \mathbb{R}_+, \tag{4.49}$$

where, as before, for any $t \in \mathbb{R}_+$,

$$\mathcal{X}^r(t) = (1 - t + \lfloor t \rfloor)\mathcal{X}^r(\lfloor t \rfloor) + (t - \lfloor t \rfloor)\mathcal{X}^r(\lfloor t \rfloor + 1).$$

Let $x^r(\cdot)$ denote the fluid scaled quantity of r^{th} system, as defined in (4.20). Then

$$\hat{x}^r(t) \quad = \quad x^r(rt). \tag{4.50}$$

In the above notation, we ignore particular randomness ω. When required we will use notation $\hat{x}^r(t,\omega)$.
For a matrix $q \in \mathbb{M}_+$, define workload vector $w(q)$ as

$$w(q) = [q_{1\cdot} \ \cdots \ q_{(n-1)\cdot} \ q_{\cdot 1} \ \cdots \ q_{\cdot(n-1)} \ q_{\cdot\cdot}].$$

Essentially, the components of $w(q)$ to $n-1$ row-sum, $n-1$ column-sum and net sum for the $n \times n$ matrix q. Intuitively, if q is a queue-size matrix for a switch, then the components of the work-load vector are: work at each of the $n-1$ input ports, work at each of the $n-1$ output ports and the total work in the switch.

Now we obtain characterization of the State Space Collapse under MWMf algorithm. The following theorem makes the precise statement.

Theorem 16. *Consider a family of IQ switch systems, indexed by $r \in \mathbb{R}_+$, satisfying Assumption 1-2, equation (4.48) and operating under the MWMf scheduling algorithm satisfying Condition 1. Let $\hat{x}^r(\cdot),\ r \in \mathbb{R}_+$ be defined as in (4.49). Then for any finite $T \geq 0$,*

$$\frac{|\hat{q}^r(\cdot) - \Delta^f(\hat{q}^r(\cdot))|_T}{|\hat{q}^r(\cdot)|_T \vee 1} \ \rightarrow \ 0, \quad \textit{in probability as } r \rightarrow \infty. \tag{4.51}$$

Here, $|\cdot|_T$ denotes sup-norm of a function defined on $[0,T]$.

The Theorem 16 motivates the following definition of State Space Collapse space of MWMf algorithms.

Definition 10 (State Space Collapse Space). *Consider a switch of size n operating under MWMf algorithm under λ such that all input and output ports are critically loaded. We call $q \in \mathbb{M}_+$ an invariant state iff $q = \Delta^f(q)$. Corresponding to an invariant state q, the workload vector $w(q) = [q_{1\cdot}, \ldots, q_{n-1\cdot}; q_{\cdot 1}, \ldots, q_{\cdot n-1}; q_{\cdot\cdot}]$, is called feasible workload vector. The space of all feasible workload ($\subseteq \mathbb{R}_+^{2n-1}$) is called the State Space Collapse Space of MWMf algorithm and it is denoted by SSC(n,MWMf).*

Theorem 16, as stated above, obtains *weak state space collapse* (see *Bramson* [1998] for definition) for all MWMf algorithm. The state space collapse is called weak, because the $|\hat{q}^r(\cdot) - \Delta^f(\hat{q}^r(\cdot))|_T$ goes to 0 on the scale of $(|\hat{q}^r(\cdot)|_T \vee 1)$. Hence, unless shown otherwise, if $|\hat{q}^r(\cdot)|_T$ grow to ∞, then it is not possible to conclude from Theorem 16 that the state of the limiting $\hat{q}^r(\cdot)$ lives in the state space collapse space. As we shall see later, this property

becomes crucial in obtaining delay optimal algorithm. Motivated by this, we state the following result for MWM-α algorithms.

Theorem 17. *Consider a family of IQ switch systems, indexed by $r \in \mathbb{R}_+$. Let the arrival process be Bernoulli IID in addition to satisfying Assumption 1-2 and equation (4.48). Further, the operates under MWM-α scheduling algorithm for $\alpha \in \mathbb{R}_+$. Let $\hat{x}^r(\cdot),\ r \in \mathbb{R}_+$ be defined as in (4.49). Then for any finite $T \geq 0$,*

$$\left|\hat{q}^r(\cdot) - \Delta^f(\hat{q}^r(\cdot))\right|_T \quad \to \quad 0, \quad \textit{in probability as } r \to \infty. \tag{4.52}$$

4.4.2 Proof of Theorem 16 on Weak State Space Collapse

To prove Theorem 16, we first establish relation between heavy traffic scaling of system and fluid scaling of system. Then we use results of section 4.2 about the equilibrium behavior of critically loaded fluid model equations to obtain the state space collapse characterization.

Heavy Traffic and Fluid Models. We wish to study the limiting process $\hat{x}(\cdot)$ over some finite time interval $[0, T]$. For the r^{th} system ($r \in \mathbb{R}_+$), the heavy traffic scaled version $\hat{x}^r(\cdot)$ is related to the fluid scaled version $x^r(\cdot)$ as

$$\hat{x}^r(t) = x^r(rt).$$

Hence to study the $\hat{x}^r(\cdot)$ on interval $[0, T]$, we define the following scaled system: for $m = 0, \ldots, \lfloor rT \rfloor$

$$x^{r,m}(t) = (a^{r,m}(t), d^{r,m}(t), q^{r,m}(t), s^{r,m}(t)),$$

where

$$a^{r,m}(t) = \frac{A^r(tz_{r,m} + rm) - A^r(rm)}{z_{r,m}} \tag{4.53}$$

$$d^{r,m}(t) = \frac{D^r(tz_{r,m} + rm) - D^r(rm)}{z_{r,m}} \tag{4.54}$$

$$s^{r,m}(t) = \frac{S^r(tz_{r,m} + rm) - S^r(rm)}{z_{r,m}} \tag{4.55}$$

$$q^{r,m}(t) = \frac{Q^r(tz_{r,m} + rm)}{z_{r,m}} \tag{4.56}$$

and

$$z_{r,m} \quad = \quad |Q^r(rm)| \vee r. \tag{4.57}$$

To study the $\hat{x}^r(\cdot)$ over finite interval $[0,T]$, that is, to study the original system $\mathcal{X}(\cdot)$ over time interval $[0,r^2T]$, we study $x^{r,m}(\cdot)$ over a finite interval $[0,L], L \geq 1$, for all $m \leq \lfloor rT \rfloor$ as this range covers the whole interval $[0,r^2T]$. The $x^{r,m}(\cdot)$ is scaled like fluid scaling. We wish to show that any limit point of $x^{r,m}(\cdot)$ as $r \uparrow \infty$ obeys fluid model equations (4.12)-(4.14) and (4.19). Now since $\lambda^r \to \lambda$ and for every $r \in \mathbb{R}_+$, r^{th} system satisfies Assumption 1, we obtain

$$\lim_{r\to\infty} a^{r,m}(t) \quad = \quad \lambda t, \qquad \text{almost surely.} \tag{4.58}$$

Given (4.58) and noting that $z_{r,m} \geq r$, it is easy to check that $x^{r,m}(\cdot)$ satisfies equations (4.29)-(4.31). This leads to the result similar to Lemma 4.

Lemma 14. *Given $\epsilon > 0$ and L, for large enough r there exists a solution of fluid model equations, $x^m(\cdot)$, such that*

$$\Pr(|x^r(\cdot) - x^m(\cdot)|_T > \epsilon) < \epsilon.$$

Next we state useful properties of $x^{r,m}(\cdot)$ as follows.

Lemma 15. *Given $\epsilon > 0$, L and T, for any $m < rT$ let $x^m(\cdot)$ be one of the limit of $x^{r,m}(\cdot)$. Then for large enough r and $T(\epsilon) \leq t \leq L$,*

$$\Pr(|q^{r,m}(t) - \Delta^f(q^{r,m}(t))| > 3\epsilon) \quad < \quad \epsilon. \tag{4.59}$$

Further, under Assumption 2,

$$\Pr(|q^{r,0}(\cdot) - \Delta^f q^{r,0}(\cdot)|_L > 3\epsilon) \quad < \quad \epsilon. \tag{4.60}$$

Proof. We first prove (4.59). From continuity of $\triangle^f$, for $\epsilon > 0$ there exists $\delta(\epsilon) > 0$ such that for any $q_1, q_2 \leq L \star \mathbf{1}$,

$$|q_1 - q_2| < \delta(\epsilon) \quad \Rightarrow \quad |\triangle^f(q_1) - \triangle^f(q_2)| < \epsilon. \tag{4.61}$$

From Lemma 14, for r large enough there exists a solution to fluid model equations, $q^m(\cdot)$, so that

$$\Pr(|q^{r,m}(\cdot) - q^m(\cdot,\omega)|_L > \min\{\epsilon, \delta(\epsilon)\}) \quad < \quad \epsilon. \tag{4.62}$$

Now, by definition $q^{r,m}(0) \leq \mathbf{1}$ and hence $q^m(0) \leq \mathbf{1}$. Hence by Lemma 13, for $t \geq T(\epsilon)$,

$$|q^m(t) - \triangle^f(q^m(t))| \quad \leq \quad \epsilon. \tag{4.63}$$

From (4.62) and (4.63) we obtain that for $t \geq T(\epsilon)$,

$$\Pr(|q^{r,m}(t) - \triangle^f(q^m(t))| > 2\epsilon) \quad < \quad \epsilon. \tag{4.64}$$

Combining (4.61), (4.62) and (4.64), we can obtain (4.59).

Next, we prove (4.60). From Assumption 2, the system starts empty, that is, $q^m(0) = \triangle^f(q^m(0)) = 0$. Hence, from (4.59) we trivially obtain (4.60).

□

Towards The Completion of Proof. Now, we use properties of $x^{r,m}(\cdot)$ to study $\hat{x}^r(\cdot)$ and obtain the proof of Theorem 16. We first state the following Lemma which is a direct consequence of Lemma 15.

Lemma 16. *Fix $\epsilon > 0$, L and T. For $r \in \mathbb{R}_+$ and $m \leq \lfloor rt \rfloor$, define $y_{r,m} = z_{r,m}/r$. Then, for large enough r*

$$\Pr(|\hat{q}^r(t) - \Delta^f \hat{q}^r(t)| > 3\epsilon y_{r,m}) \quad < \quad \epsilon, \tag{4.65}$$

for

$$\frac{y_{r,m}T(\epsilon) + m}{r} \leq t \leq \frac{Ly_{r,m} + m}{r}.$$

Further, under Assumption 2,

$$\Pr(|\hat{q}^r(\cdot) - \Delta^f \hat{q}^r(\cdot)|_{\bar{L}} > 3y_{r,0}\epsilon) \quad < \quad \epsilon, \tag{4.66}$$

where $\bar{L} = Ly_{r,0}/r$.

Proof. The proof follows from Lemma 15. From definition

$$q^{r,m}(t) = \frac{1}{y_{r,m}} \hat{q}^r \left(\frac{t y_{r,m} + m}{r} \right). \tag{4.67}$$

The Condition 1 regarding weight function $f(\cdot)$ and the structure of the optimization problem $\Delta^f(\cdot)$ implies

$$\Delta^f(\alpha q) = \alpha \Delta^f q, \quad \text{for any} \quad \alpha \in \mathbb{R}_+. \tag{4.68}$$

Now the statement of Lemma 16 follows from (4.67), (4.68) and Lemma 15. □

Next, we use Lipschitz property of $q^{r,m}(\cdot)$ to obtain a bound on the rate at which $y_{r,m}$ can increase.

Lemma 17. *For $r \in \mathbb{R}_+$ and $m \leq \lfloor rT \rfloor$*

$$y_{r,m+1} \quad \leq \quad 2y_{r,m}. \tag{4.69}$$

Proof. From definition (see (4.57)),

$$y_{r,m} = z_{r,m}/r \geq 1.$$

By the property of a switch that at most one arrival can happen to a queue in a given time slot, we obtain the following.

$$\begin{aligned} y_{r,m+1} &= \frac{Q^r(rm+r)}{r} \vee 1 \\ &\leq \frac{Q^r(mr)+r}{r} \vee 1 \\ &\leq \left(\frac{Q^r(mr)}{r} \vee 1 \right) + 1 \\ &= y_{r,m} + 1 \\ &\leq 2y_{r,m}. \end{aligned} \tag{4.70}$$

□

For a given $t \in [0,T]$ and $r \in \mathbb{R}_+$, define $m_r(t)$ as follows:

$$m_r(t) \quad = \quad \arg\min_{m \geq 0} \left\{ \frac{m}{r} \leq t \leq \frac{Ly_{r,m} + m}{r} \right\}. \tag{4.71}$$

Next we obtain an estimate on $m_r(t)$.

Lemma 18. *Fix $\epsilon > 0, L$, and T and $\delta \leq 1$. Then for large enough r and $t \in \left[\frac{\delta L y_{r,0}}{r}, T\right]$,*

$$rt - m_r(t) \quad \geq \quad \frac{\delta L y_{r,m_r(t)}}{2}. \tag{4.72}$$

Proof. For $t \in \left[\frac{\delta L y_{r,0}}{r}, \frac{L y_{r,0}}{r}\right]$, by definition $m_r(t) = 0$, which satisfies (4.72). For $t > L y_{r,0}/r$, it follows that $m_r(t) \geq 1$. By definition of $m_r(t)$, we obtain

$$rt - (m_r(t) - 1) \quad > \quad L y_{r,m_r(t)-1}. \tag{4.73}$$

From Lemma 17,

$$y_{r,m_r(t)} \quad \leq \quad 2 y_{r,m_r(t)-1}. \tag{4.74}$$

From (4.73) and (4.74), we obtain

$$rt - m_r(t) \quad > \quad \frac{L y_{r,m_r(t)}}{2}. \tag{4.75}$$

This completes the proof of Lemma 18. □

Now we are ready to complete the proof of Theorem 16.

Proof. (Theorem 16.) Let T be given. Then for any $\epsilon > 0$, choose L satisfying

$$L > \frac{2T(\epsilon)}{\delta}.$$

From Lemma 18 and given that $L > \frac{2T(\epsilon)}{\delta}$ for all $t \in \left[\frac{\delta L y_{r,0}}{r}, T\right]$,

$$\begin{aligned} rt - m_r(t) \quad &\geq \quad \frac{\delta L y_{r,m_r(t)}}{2} \\ &\geq \quad T(\epsilon) y_{r,m_r(t)}. \end{aligned} \tag{4.76}$$

From definition

$$|\hat{q}^r(\cdot)|_T \vee 1 \quad \geq \quad y_{r,m}, \quad \forall m. \tag{4.77}$$

From (4.76),(4.77) and Lemma 16 and we obtain

$$\Pr(|\hat{q}^r(t) - \triangle^f(\hat{q}^r(t))| > 3\epsilon(|\hat{q}^r(\cdot)|_T \vee 1)) \quad < \quad \epsilon, \tag{4.78}$$

for $t \in [\delta L y_{r,0}/r, T]$. Further, when Assumption 2 holds, by Lemma 16 the (4.78) holds for $t \in \left[0, \frac{Ly_{r,0}}{r}\right]$. Now, $[0, Ly_{r,0}/r] \cup [\delta L y_{r,0}/r, T] = [0, T]$. This completes the proof of Theorem 16.

□

4.4.3 Proof of Theorem 17: on Strong State Space Collapse

Now, we prove Theorem 17 using some results of Chapter 2 and Theorem 16. Now, in order to prove Theorem 17, given the result of Theorem 16, we only need to show that $\lim_{r\to\infty} |\hat{q}^r(\cdot)|_T = O(1)$ in probability. For ease of exposition, we present arguments for $\alpha = 1$. Exactly the same arguments will work for any positive finite $\alpha \in \mathbb{R}_+$.

Consider the case when $\alpha = 1$. Consider r^{th} system for some large r. Lets go back to original time-scale from heavy traffic scaling. Consider time interval $[0, \lceil r^2 T \rceil]$. The arrival rate to the system is $\lambda(r) = \lambda - \frac{1}{r}\Phi$. Hence, the maximal net load is $\lambda^*(r) = 1 - \Theta(1/r)$. Let the $Q^r(m)$ denote the queue-size matrix at time $m \in [0, \lceil r^2 T \rceil]$. To show, $|\hat{q}^r(\cdot)|_T$ is $O(1)$, it is sufficient to show that the maximum queue-size attained in the interval $[0, \lceil r^2 T \rceil]$ is $O(r)$ (see definition (4.49)). Hence, next we show that under Bernoulli IID traffic with arrival rate-matrix $\lambda(r)$ such that $\lambda^*(r) = 1 - \Theta(1/r)$, the maximum queue-size at any queue is $O(r)$ with probability $1 - o(1)$ (where probability scaling is in terms of r).

Recall proof of Theorem 1 of Chapter 2. The proof used quadratic Lyapunov function, $L(Q(m)) = \sum_{i,j} Q_{ij}^2(m)$. (Here, we drop reference to r in the notation $Q^r(m)$ so as to avoid possible confusion between exponent 2 and index r.) The inequality (2.16) is reproduced here as follows.

$$E[L(Q(m+1)) - L(Q(m))|Q(m)] \quad \leq \quad -2\frac{(1-\lambda^*)}{n}\|Q(m)\|_1 + 2n. \tag{4.79}$$

Let $Y(m) = L(Q(m))$. Now, by non-negativity of each of $Q_{ij}(m)$,

$$\sum_{i,j} Q_{ij}(m) \quad \geq \quad \left(\sum_{i,j} Q_{ij}^2(m) \right)^{0.5}. \tag{4.80}$$

Now, (4.79), (4.80), $(1 - \lambda^*(r)) = \Theta(1/r)$ and notation $Y(m) = L(Q(m))$, give us the following.

$$E[Y(m+1)] \quad \leq \quad E[Y(m)] - \frac{2}{nr}\sqrt{Y(m)} + 2n. \tag{4.81}$$

Now, ignoring the addition term in (4.81), essentially $Y(m)$ is a positive super-martingale (for technical completeness, one can define precise super-martingale as $X(m) = Y(m)\mathbf{1}_{\{Y(m)>2n^4r^2\}} + 2n^4r^2\mathbf{1}_{\{Y(m)\leq 2n^4r^2\}}$). Hence, by Dubin's inequality(see Chapter 4, *Durrett* [1995]) for upcrossing of interval $[100n^4r^2, K100n^4r^2]$ (applied to super-martingale) starting from $Y(0) = 0$, gives us that the number of upcrossing is at least 1 with probability at most $1/K$. That is, given $\epsilon > 0$, the maximum value of $Y(m)$ over interval $[0, \lceil r^2T \rceil]$ is no more than $\frac{100n^4r^2}{\epsilon}$ with probability at least $1 - \epsilon$. That is,

$$\Pr\left(\max_{0\leq m\leq \lceil r^2T \rceil} Y(m) = O(r^2) \right) \quad \geq \quad 1 - \epsilon, \tag{4.82}$$

for any $\epsilon > 0$. By definition, $Y(m) = \sum_{i,j} Q_{ij}^2(m)$. From the well-known relation between ℓ_2 and ℓ_1 norm, $n^2 Y(m) \geq \sum_{i,j} Q_{ij}(m)$. Hence, we obtain that

$$\Pr\left(\max_{0\leq m\leq \lceil r^2T \rceil} \sum_{i,j} Q_{ij}(m) = O(r) \right) \quad \geq \quad 1 - \epsilon, \tag{4.83}$$

This in turn implies that,

$$\Pr\left(|\hat{q}^r(\cdot)|_T = O(1) \right) \quad \geq \quad 1 - \epsilon. \tag{4.84}$$

This completes the proof of Theorem 17 for $\alpha = 2$. The main ingredient used in this proof is the Lyapunov drift equation to obtain super martingale. Such Lyapunov drift is available for all MWM-α by design. Hence, using arguments as above, Theorem 17 can be proved for all $\alpha \in \mathbb{R}_+$.

4.5 Inferring Performance via State Space Collapse

The Theorem 16 suggests that under heavy traffic scaling, the scaled version of the system is always in an MWMf endstate. That is, given input and output workload, the queue-sizes are determined by the Lifting Map, $\Delta^f(\cdot)$. Thus, in order to determine state of the switch, it is sufficient to track the input and output workload vectors. This simplicity in the description of the system opens up the possibility of making more refined statement about the performance of algorithm. To explain this subtle issue, we review some of the well known techniques and their failure to study performance of scheduling algorithm. Then, we will use the state space collapse property of MWM-α algorithm to obtain the characterization of a delay optimal algorithm (at the heavy traffic scale). We find that MWM-α, as $\alpha \to 0^+$, is an optimal algorithm. We obtain description of this algorithm at the actual time scale and find it very similar to the Longest Port First algorithm proposed by *Mekkittikul and McKeown* [1998]. We also find that MWM-1 algorithm is not optimal. Finally, we use the state space collapse characterization to provide an explanation of Conjecture 1 of Chapter 1.

4.5.1 Failure of Known Methods

A large body of literature has been developed for more than past 40 years to understand performance of queueing systems or networks in a stochastic setting. The motivation of analyzing networks in most generality has led to a beautiful development of stochastic networks theory. The tools developed in stochastic networks theory have been successful in many situations to analyze performance of system in terms of throughput (e.g. fluid model technique, Lyapunov function theory, etc.) and delay (e.g. queueing theory, theory of large deviations, etc.).

For the switch system, traditional methods like Lyapunov function theory and fluid model technique have been successful as shown in this thesis in the chapters 2, 3 and 4 till now. Thus, as far as throughput performance of algorithms is concerned, traditional approaches have been extremely successful.

The delay analysis of switch is not well understood. We obtained bounds on average delay in chapter 2 with the help of Lyapunov functions for Bernoulli IID arrival process. Unfortunately, as shown in section 2.3 of chapter 2, these bounds are not tight. Hence, they do not allow comparison of algorithms based on delay performance nor allow characterization of optimal algorithm.

A standard queueing theory approach is useful to analyze delay of a queue when both arrival and service distributions are known. In the case of switch, arrival process is external and hence well known. But, the service distribution for any queue strongly depends on the scheduling algorithm and the decision of scheduling algorithm depends on the whole system. This makes it impossible to characterize the service distribution induced by the algorithm. Hence, standard approach does not work to analyze queueing delay.

In the context of ATM networks, theory of Large Deviations has been extremely successful (see books by *Dembo and Zeitouni* [1998] for theory of Large Deviations and book by *Ganesh et al.* [2004] for its application to the queueing systems). The main reason for the success was the possibility of decoupling large systems into small system. For example, in case of an n port Output Queued switch, the system can be seen as made of n independent single FIFO queues with deterministic service rate. Hence, such system can be analyzed. For Input Queued switch, due to dependencies induced by algorithm, such decomposition is not possible.

In stochastic networks, the tool of stochastic coupling has been very well exploited to compare performance of two systems. Such results do not characterize exact performance but provide relative behavior. Though the results are weaker than exact performance characterization, they can be possibly useful in context of switch due to their generality of application. However, obtaining such coupling arguments in the context of switch requires one to study the structure of the system in a great detail. We find it very difficult to apply directly on the actual system.

Instead, in the subsequent sections, we apply a modified stochastic coupling to characterize optimal algorithm as well as compare performance of algorithms by looking at the system in the heavy traffic scale. The system in heavy traffic are easy for this purpose is purely because of their state space collapse property or equivalently possibility of describing the complete state of the system only via input-output workload vectors.

4.5.2 An Optimal Algorithm

In this section, we characterize an optimal algorithm at the heavy traffic scale. For this purpose, we will focus on studying state space collapse of MWM-α algorithms. We assume that all input and output ports are critically loaded. That is,

$$\lambda_{i\cdot} = \lambda_{\cdot j} = 1, \quad 1 \leq i, j \leq n.$$

For simplicity of notation, in the rest of the section, we use $q(t)$ in place of $\hat{q}(t)$. Now, we define an optimal algorithm at the heavy traffic scale.

Definition 11 (Optimal Algorithm). *An algorithm $\mathcal{A}$ is called optimal at heavy traffic scale if under identical arrivals the scaled workload vector $w(q(t))$ is component-wise no more than the scale workload vector of any other algorithm.*

Next, we state the following characterization of an optimal algorithm.

Theorem 18. *The limiting algorithm $\lim_{\alpha\to 0_+}$ MWM-α is an optimal scheduling algorithm in the sense of Definition 11 for any $n \times n$ switch.*

To prove the Theorem 18, we will require some Lemmas. Let the limiting algorithm $\lim_{\alpha\to 0_+}$ MWM-α be denoted by $\mathcal{A}^\star$.

We recall some notations before presenting next few Lemmas. In the context of $n \times n$ switch, let $q \in \mathbb{M}_+$ be the queue-size and $w(q) \in \mathbb{R}_+^{2n-1}$ be corresponding workload vectore, where

$$w_{2n-1}(q) = q_{\cdot\cdot},$$

$$w_i(q) = q_{i\cdot},\ 1 \le i < n, \quad \text{and}$$

$$w_{j+n-1} = q_{\cdot j},\ 1 \le j < n.$$

Now, we state the Lemma about State Space Collapse characterization of $\mathcal{A}^\star$.

Lemma 19. *For any $n \times n$ switch, the state space collapse space of algorithm $\mathcal{A}^\star$ is a complete space, that is,*

$$SSC(n, \mathcal{A}^\star) \;=\; \{w = (w_1, \ldots, w_{2n-1}) \in \mathbb{R}_+^{2n-1} \;:\; w_i > 0,\ \forall i\}. \tag{4.85}$$

Proof. We prove this by contradiction. Suppose the statement of Lemma is not true. That is, there exists a workload vector $w = (w_1, \ldots, w_{2n-1})$ satisfying conditions of (4.85) which is not feasible as defined in Definition 10. That is, for any matrix $q \in MatriceP$ with $w = w(q)$

$$q \neq \Delta^{\mathcal{A}^\star}(w(q)),$$

where $\Delta^{\mathcal{A}^\star}(\cdot)$ denotes the lifting map of algorithm of $\mathcal{A}^\star$. As shown before, there exists a solution to the convex optimization problem $q^* = \Delta^{\mathcal{A}^\star}(w)$ (for ease of understanding, treat $\mathcal{A}^\star$ as an MWM-α algorithm with a fixed but very small α.)

Now, $w(q^*) \neq w$. In particular, it must be larger than w in at least two of the components (one row and one column). Without loss of generality, let $w(q^*)_1 > w_1, w_n^* > w_n$ and $w(q^*)_i \geq w_i$ otherwise. Now it must be the case that $q^*11 = 0$. If not, then we can reduce q_{11}^* either till it becomes 0 or $w_1^* = w_1$ or $w_n^* = w_n$. Thus, $q_{11}^* = 0$. Now due to w being positive in all components, there exists i, j such that $q_{1i}^*, q^*j1 > 0$. Without loss of generality, let $i = j = 2$. Now, since q^* satisfies the convex optimization problem, $\mathcal{A}^\star$-CP(q^*) corresponding to the MWM-0^+-CP($\cdot$), by Theorem 15 it must be the case that the weight of all matchings are equal.

Under algorithm $\mathcal{A}^\star$, the weight of entry (i, j) is $\lim_{\alpha \to 0_+} \left(q_{ij}^*\right)^\alpha$. Now for very small α,

$$\left(q_{ij}^*\right)^\alpha \approx 1 + \alpha \log q_{ij}^*. \tag{4.86}$$

Thus, for $\alpha \to 0_+$, essentially the weight is 1 if entry is non-zero and zero otherwise. Now consider two matchings: π and $\hat{\pi}$ where $\pi(k) = k, \forall k$, while $\hat{\pi}(1) = 2$; $\hat{\pi}(2) = 1$; $\hat{\pi}(k) = k, k \geq 3$. Then, it is easy to see that the weight of π is strictly smaller than the weight of $\hat{\pi}$ since $q_{11}^* = 0$ while $q_{12}^*, q_{21}^* > 0$. This is a contradiction.

Thus, the original assumption of $w(q^*) \neq w$ is false. That is, w is a feasible workload vector under algorithm $\mathcal{A}^\star$. This completes the proof of Lemma 19. □

As an immidate corollary of Lemma 19, we obtain the following (which we state as a Lemma).

Lemma 20. *Under $\mathcal{A}^\star$ algorithm, let q be an invariant state. Then,*

$$q_{ij} > 0 \quad \Leftrightarrow \quad q_{i\cdot} > 0 \text{ and } q_{\cdot j} > 0. \tag{4.87}$$

Proof.
($\Rightarrow$) This is a straightforward implication: if $q_{ij} > 0$ then $q_{i\cdot}, q_{\cdot j} > 0$.
($\Leftarrow$) This follows using very similar arguments as used to prove Lemma 19. □

Lemma 21. *Let q be such that all input workloads and output workloads are non-zero, that is,*

$$q_{i\cdot} > 0, \quad \forall i; \qquad q_{\cdot j} > 0, \quad \forall j. \tag{4.88}$$

Then, under $\mathcal{A}^\star$ all input and output workloads are served at unit rate.

Proof. From Lemma 20, under (4.88), all entries are strictly positive. Hence, whatever matching $\mathcal{A}^\star$ chooses to serve, its never going to idle. Hence, by the property of matching, each input is served at unit rate as well as each output is served at unit rate. □

Lemma 22. *Under heavy traffic scaling, for any scheduling algorithm, the limiting queue-sizes are such that all input and output workloads are non-zero with probability* 1.

Proof. Consider any input i. Under heavy traffic scaling, the limiting arrival process has rate 1. Under any scheduling algorithm, the net service rate is at most 1. Thus, workload at input i, q_i. can be lower bounded by that of an $\cdot/D/1$ queue with deterministic service of rate 1. The well known results in queueing theory imply that the queue-size of such a queue under heavy traffic scaling (equivalently, when arrival rate is 1) becomes a reflected Brownian motion. For such reflected Brownian motion, the set of time when it is 0 is measure 0. That is, with probability 1, the queue-size of such a queue is non-zero. That is, the workload at input i is non-zero with probability 1.

The similar argument applies for all output workload. This completes the proof of Lemma 22. □

Proof of Theorem 18. Consider an $n \times n$ switch under heavy traffic scaling. By Lemma 22, under any algorithm the input and output workloads are non-zero with probability 1. Given the switch constraints, no algorithm can serve input workload or output workload at rate more than 1. In particular, from Lemma 21, $\mathcal{A}^\star$ serves all input and output workloads at rate 1 with probability 1. Hence, the input and output workload are minimal under $\mathcal{A}^\star$ algorithm at all the time under heavy traffic scaling. This completes the proof of Theorem 18. □

4.5.3 MWM-1 is Not Optimal

This section is dedicated to the following theorem, stating that MWM-1 is not optimal.

Theorem 19. *The algorithm MWM (i.e. MWM-1) is not optimal.*

To prove the Theorem 19, we need the following state space collapse characterization of MWM-1.

Lemma 23. *For any $n \times n$ switch,*

$$\begin{aligned} \textit{SSC(n,MWM-1)} \;=\; \{w \in \mathbb{R}_+^{2n-1} :\; & w_i + w_{j+n-1} \geq \frac{w_{2n-1}}{n}, \;\; 1 \leq i,j \leq n-1; \\ & \frac{(n-1)w_{2n-1}}{n} \geq \sum_{k=1}^{n-1} w_{k+n-1} - w_i, 1 \leq i \leq n-1; \\ & \frac{(n-1)w_{2n-1}}{n} \geq \sum_{k=1}^{n-1} w_k - w_{j+n-1}, 1 \leq j \leq n-1; \\ & w_k \leq w_{2n-1}, \;\; 1 \leq k \leq 2n-1\}. \end{aligned} \tag{4.89}$$

Proof. Let $w \in \mathbb{R}_+^{2n-1}$ be a workload vector for a $n \times n$ switch. For its feasibility under MWM-1, there must exists a $n \times n$ positive matrix $q \in \mathbb{M}_+$ such that

$$\Delta^{\alpha}(q) = q \;\; \text{and for } 1 \leq i \leq n-1, \;\; q_{i\cdot} = w_i, \;\; q_{\cdot i} = w_{i+n-1}, q_{\cdot\cdot} = w_{2n-1}.$$

This implies that given net-work w_{2n-1}, for any feasible w, the corresponding invariant $q \geq (0)$. Thus, to characterize SSC(n,MWM-1), we need to characterize q in terms of w and obtain the conditions for it being a positive matrix.

Given w, from Theorem 15, if $\lambda > 0$ component-wise, the invariant state has the property that all matchings are of equal weight. Given workload vector w, the weight of each matching will be w_{2n-1}/n. Now, a simple computation will lead to the following positivity characterization

$$q_{ij} \geq 0 \quad \Leftrightarrow \quad q_{i\cdot} + q_{\cdot j} \geq \frac{w_{2n-1}}{n}, \;\; \forall \; i,j. \tag{4.90}$$

Since q is an invariant state corresponding to the workload vector w, it must be that $w(q) = w$. Hence, by definition $q_{i\cdot} = w_i$ and $q_{\cdot j} = w_{j+n-1}$ for $1 \leq i,j \leq n-1$; $q_{n\cdot} = w_{2n-1} - \sum_{k=1}^{n-1} w_k$ and $q_{\cdot n} = w_{2n-1} - \sum_{k=1}^{n-1} w_{k+n-1}$. Now, replacing these in (4.90) essentially gives the characterization of SSC(n,MWM-1) as described in (4.89).

This completes the proof of Lemma 23. □

Proof of Theorem 19. The Lemma 23 suggests that the SSC(n,MWM-1) is a strictly smaller sub-space of $\mathbb{R}_+^{2n-1}$. There exists arrival process such that under algorithm $\mathcal{A}^{\star}$, the workload vector can take value outside of SSC(n,MWM-1). The MWM-1 algorithm, in such conditions will retain its workload vector inside SSC(n,MWM-1) by idling on some port. Thus, losing

performance. This proves that MWM-1 algorithm will not be optimal as defined in Definition 11. This completes the proof of 19. □

4.5.4 An Explanation of Conjecture 1

In this section, we offer an explanation for the Conjecture 1. In order to do so, we study the state space collapse space of MWM-α algorithms. We first state result comparing the state space collapse space of MWM-α algorithms for 2×2 switches. We strongly believe that the following result hold in general for any $n \times n$ switch.

Lemma 24. *For $\alpha_1, \alpha_2 \in \mathbb{R}_+$, if $\alpha_1 < \alpha_2$ then*

$$SSC(2, \textit{MWM-}\alpha_2) \quad \subseteq \quad SSC(2, \textit{MWM-}\alpha_1). \tag{4.91}$$

Proof. Let $w = (w_1, w_2, w_3) \in \mathbb{R}^3_+$ be a workload vector for a 2×2 switch. For its feasibility under MWM-α, there must exists a 2×2 positive matrix $q \in \mathbb{M}_+$ such that

$$\Delta^\alpha(q) = q \quad \text{and} \quad q_{1\cdot} = w_1, \; q_{\cdot 1} = w_2, q_{\cdot\cdot} = w_3.$$

This implies that given net-work w_3, for any feasible w, $w_1, w_2 \leq w_3$ and the corresponding invariant $q \geq (0)$. Now $q \geq (0)$ further constraints the possible values w_1, w_2 can take, given w_3. Hence to characterize SSC(2,MWM-α), we need to first characterize q in terms of w and obtain the conditions for it being positive matrix.

Given w, from Theorem 15, if $\lambda > 0$ component-wise, the invariant state has the property that both matchings are of equal weight, where weight is α^{th} power of queue-size. Given workload vector w, the input workloads are $w_{1\cdot} = w_1$ and $w_{2\cdot} = w_3 - w_1$ while output workloads are $w_{\cdot 1} = w_2$ and $w_{\cdot 2} = w_3 - w_2$. This leads to the following positivity characterization

$$q_{ij} \geq 0 \quad \Leftrightarrow \quad w_{i\cdot} + w_{\cdot j} + \left(w_{i\cdot}^\alpha + w_{\cdot j}^\alpha\right)^{1/\alpha} \geq w_3. \tag{4.92}$$

Thus, inequalities on the right hand side of (4.92) characterize the SSC(2,MWM-α). Now, consider the following known analysis result.

Lemma 25. *For any $x, y \in \mathbb{R}_+$ and any $\theta \geq 1$,*

$$\left(x^\theta + y^\theta\right)^{1/\theta} \quad \leq \quad x + y. \tag{4.93}$$

Now consider $0 < \alpha_1 < \alpha_2$. Then, for any $a, b \in \mathbb{R}_+$,

$$(a^{\alpha_2} + b^{\alpha_2})^{1/\alpha_2} \quad \leq \quad (a^{\alpha_1} + b^{\alpha_1})^{1/\alpha_1}. \tag{4.94}$$

The (4.94) follows from Lemma 25 by taking $x = a^{\alpha_1}$ and $y = b^{\alpha_1}$. From (4.92) and (4.94), it is easy to conclude that

$$SSC(2, \text{MWM-}\alpha_2) \subseteq SSC(2, \text{MWM-}\alpha_1).$$

This completes the proof of the Lemma 24. □

The Lemma 24 suggests that as α increases the state space collapse space decreases. Now the state of the system (i.e. workload vector) roams inside the state space collapse space. Every time it hits the boundary, the switch algorithm selects matching so that the state of the system remains inside the state space collapse space by idling on some port. This intuitively means that given the same arrival process, the switch is more likely to idle for smaller state space collapse space. Hence, from the result of Lemma 24, the switch performance should become worse as α increases under the MWM-α algorithm. This offers an intuitive explanation to the Conjecture 1 for a 2×2 switch. Next, we make this intuition rigorous.

Ideally, we would like to obtain the result of the following type: the workload of algorithm MWM-α_1 is dominated by the workload of MWM-α_2 algorithm for $\alpha_1 < \alpha_2$ under heavy traffic scaling. Suppose the following was true: *the feasibility of all input workloads only depended on the value of other input workloads (similarly for output workload)*. Then, using statement of Lemma 24, obtaining the ideal result is a straightforward coupling.

Unfortunately, as shown in Lemma 24, the state space collapse characterization is such that feasibility of an input workload depends on other input as well as output workloads. Hence, obtaining a coupling is very hard. Hence, in order to compare algorithms, we consider a specific arrival process with arbitrary starting position. We describe the setup next.

Consider a 2×2 switch. Let the arrival process be determistic with rate λ such that all ports are critically loaded and $\lambda_{ij} > 0$. For 2×2 switch, such a λ can be written as

$$\lambda = a\pi_1 + (1-a)\pi_2, \ a \in (0,1),$$

where π_1 serves queues (1,1) and (2,2) while π_2 serves queues (1,2) and (2,1). Let the initial state of the switch be any $w \in \mathbb{R}^3_+$. Now, let the switch be operating under algorithm MWMf.

If $w \in$ SSC(2,MWMf) then the intial switch-state q is such that $q = \Delta^f(q)$ and $w(q) = q$. If $w \notin$ SSC(2,MWMf), then q is the solution to the following optimization problem.

$$\min_{q' \in \mathbb{M}_+} \max_{\pi \in \mathbb{P}} \pi \cdot f(q')$$

$$\text{such that} \quad w(q') = w.$$

Now, we state the following theorem comparing MWM-α algorithms.

Theorem 20. *Under the setup described above, the workload vector under algorithm MWM-α_1 is component-wise dominated by the workload vector under algorithm MWM-α_2 for $0 < \alpha_1 < \alpha_2$.*

Proof. For ease of exposition, the proof if presented for $\alpha_1 = 1$ and $\alpha_2 = 2$. The arguments can be easily extended for any $0 < \alpha_1 < \alpha_2$.

Consider a $w \in \mathbb{R}^3_+$. From Lemma 24, there are three possibilities:
(1) $w \in$ SSC(2,MWM-1) and $w \in$ SSC(2,MWM-2).
(2) $w \in$ SSC(2,MWM-1) and $w \notin$ SSC(2,MWM-2).
(3) $w \notin$ SSC(2,MWM-1) and $w \notin$ SSC(2,MWM-2).

In what follows, we consider the situation where $w_1 \leq w_2 \leq w_3/2$. All other (total 8) situations can be reduced to this by renumbering input/output and changing input/output definition. Let $u(t)$ and $v(t)$ denote the workloads at time t under algorithms MWM-1 and MWM-2 respectively, with $u(0) = v(0) = w$. Recall that both receive arrivals at deterministic rate $\lambda = a\pi_1 + (1-a)\pi_2, a \in (0,1)$.

Case (1). In this case, the switch starts in the invariant state for both MWM-1 and MWM-2. Hence, by Theorem 14, it remains in the same state forever, that is, $u(t) = v(t) = w$ for all $t \geq 0$.

Case (2). In this case, the switch starts in the invariant state for MWM-1 and hence $u(t) = w$ for all $t \geq 0$. On the contrary, for MWM-2, the switch starts in non-invariant state. As per above setup, the intial switch state q corresponding to w is such that it has a unique maximum weight matching. Due to $w_1 \leq w_2 \leq w_3/2$, π_1 will be the maximum weight matching and corresponding initial state has $q_{11} = 0$. Now, MWM-2 will serve π_1 at unit rate till both matchings become of equal weight. During this time, (i) MWM-2 idles at q_{11} for $(1-a)$ fraction of the time since $\lambda = a\pi_1 + (1-a)\pi_2$, (ii) MWM-2 increases $v_1(t), v_2(t)$ at rate

$(1-a)$, and (iii) MWM-2 increases $v_3(t)$ at rate $1-a$. Now, on reaching invariant state, the MWM-2 retains this invariant state from then on. Thus, under this case, $u(t) < v(t)$ for all $t > 0$.

Case (3). In this case, both algorithms start with initial state that is non-invariant. Both algorithms have π_1 as unique maximum weight matching in this initial state and both algorithms serve π_1 at unit rate till they reach invariant state. During this time, both algorithms (i) idle at q_{11} for $(1-a)$ fraction of the time since $\lambda = a\pi_1 + (1-a)\pi_2$, (ii) they increase $u_1(t), u_2(t), v_1(t), v_2(t)$ at rate $(1-a)$, and (iii) increases $u_3(t), v_3(t)$ at rate $1-a$. Thus, given $u(0) = v(0) = w$, both algorithms change their workload $u(t), v(t)$ in the same direction. Now, by Lemma 24,

$$\text{SSC(2,MWM-2)} \subset \text{SSC(2,MWM-1)}.$$

Hence, it must be the case that $u(t)$ reaches SSC(2,MWM-1) quicker than $v(t)$ reaching SSC(2,MWM-2). Let T_1 be the time when $u(t)$ reaches SSC(2,MWM-1). Then, we obtain that, $u(t) = v(t), \quad t \leq T_1$, and $u(t) < v(t), \quad t > T_1$.

Thus, as shown in cases (1), (2) and (3), $u(t) \leq v(t)$, for all $t \geq 0$ under any initial workload $w \in \mathbb{R}^3_+$. This completes the proof of Theorem 20. □

4.6 Chapter Summary and Discussion

This chapter was dedicated to the study of throughput and delay property of generalized Maximum Weight Matching algorithm, denoted by MWMf.

We obtained characterization of all stable weight functions f. We used fluid model techniques and Lyapunov functions theory to analyze throughput of MWMf algorithms. The throughput results suggest that a large class of algorithms provide optimal throughput.

This naturally led us to the following question:*which, among all of these throughput optimal MWMf, is a delay optimal algorithm?*. The traditional methods failed in answering this question. In search of an answer to the above question, we studied the IQ switch under heavy traffic scaling. We obtained the state space collapse property for MWMf algorithm via fixed points of equilibrium fluid model equations. As an aside, we note that the results on equilibrium fluid model equations revealed interesting properties of these matching algorithms.

Interestingly, the state space characterization of MWM-0^+ algorithm allowed us to prove its optimality. Now, the description of MWM-α algorithm for all $\alpha > 0$ remains the same for the system operating at the heavy traffic scale (or fluid scale) and the discrete scale.

The approximation (4.86) suggests that, MWM-0^+ must do the following: *among all possible maximum size matching, choose the maximum weight (weight is logarithm of queue-size) maximum size matching*. Now, at discrete scale, the queue-sizes are always integer. Further, if queue-sizes are assumed to be bounded above by some constant, then there exists a small enough α such that the above description becomes exact.

This also reminds us of the Longest Port First (LPF) algorithm propose by *Mekkittikul and McKeown* [1998]. The LPF algorithm chooses the Maximum Weighted Maximum Size Matching where weight of an edge (i,j) is the sum of the workload at input i and output j. Based on this, we believe that the Longest Port First algorithm is an optimal algorithm. The difficulty in proving this statement is of the technical form: the description of LPF is not easy for fluid model analysis as it involves modeling Maximum Size Matching.

In addition to identifying the optimal algorithm, we used the method to demonstrate that the usual Maximum Weight Matching algorithm is not optimal. This falsified one of the long standing folk-fore in the switching community. We also used the methods to provide explanation to the observation of by *Keslassy and McKeown* [2001a] noted as Conjecture 1 in the beginning of the thesis.

We believe that the method of this chapter are quite general. In particular, we believe that methods can be extended to a large class of scheduling problems where "MWM-type" algorithms are throughput optimal algorithms. For example, framework of Radio-hop network used by *Tassiulas and Ephremides* [1992]. In general, the results of this chapter leads to the following intuitive understanding of optimality of algorithms: an optimal algorithm is the one that has maximal state space collapse space so as the idling in the system is minimized.

4.7 Bibliographic Notes

A part of results of Section 4.2 are published by *Shah* [2001]. The results of the Sections 4.3, 4.4 and 4.5 are part of a preprint by *Shah and Wischik*. These results motivated by companion papers by *Bramson* [1998] and *Williams* [1998].

The fluid model for a switch was first developed by *Dai and Prabhakar* [2000]. They used the fluid model to analyze throughput of MWM and Maximal Matching algorithms. Fluid model technique has been very well developed and used in various context. See noted by *Dai* [1999] for a detailed exposition on this subject. The work by *Stolyar* [2004] studied a input queued type switch under heavy traffic. This work restricts the number of port that are

critically loaded to one. This, in turn, obtains one-dimensional state space collapse space for all algorithms and hence can not differentiate performance of MWMf algorithms for different weight functions.

The state space collapse phenomenon was first observed by *Whitt* [1971]. A series of results were obtained on heavy traffic analysis of basic queueing systems in the early 1970s, notably by *Iglehart and Whitt* [1970a], *Iglehart and Whitt* [1970b], *Iglehart and Whitt* [1971]. This led to a wonderful development of theory of heavy traffic for complex queueing systems. For example, works by *Harrison* [1988], *Harrison* [1995], *Harrison and Williams* [1992] and *Reimann* [1984]. A good reference for the early development of the heavy traffic theory is the book by *Harrison* [1985]. The results of *Bramson* [1998] and *Williams* [1998] have provided a standard technique to obtain state space collapse characterization of systems under heavy traffic scaling. They pioneered the use of equilibrium fluid model to obtain the state space collapse property.

CHAPTER 5

Conclusions and Future Work

This thesis was about desing and analysis of scheduling algorithms for the IQ switches. The memory bandwidth requirement is becoming a major bottleneck in designing high-speed switches. Due to low memory bandwidth requirement, the IQ switch architecture is currently very popular for designing high speed switches. But, IQ switches require good scheduling algorithm in order to provide good performance. Implementation concerns make simple algorithms desirable. But if algorithm is too simple, it may perform rather poorly. Thus, one is required to resolve the tension between implementability and performance of scheduling algorithm.

Motivated by this challenge, one part of this thesis (Chapter 3) provided a suite of simple to implement high performance scheduling algorithms – APSARA, LAURA and SERENA. These algorithms were based on novel design ideas like (i) use of information from past, (ii) use of arrival information, (iii) exploiting problem structure (Merge procedure) and (iv) use of parallelism for search in the space of matchings; along with the well-known technique of randomization. We proved that the proposed algorithms provide 100% throughput and have low delay. Our simulations showed that they perform very competitively relative to known good algorithm, MWM. We discussed the implementation details of this algorithm and find that algorithms like APSARA and SERENA are implementable in current switches in core-routers.

The second part of this thesis presented novel analysis methods for scheduling algorithms. In Chapter 2, we analyzed throughput and delay property of MWM and its approximations using a method based on Lyapunov functions. These methods, though applicable to Bernoulli IID traffic only, provide a great insight and useful bounds on performance of algorithms. Motivated by the consideration of general distributions for arrival process, in the Section 4.2 of Chapter 4, we used fluid models to analyze throughput of algorithms. In particular, we showed that a large class of MWM-type algorithms have optimal throughput. But, they have different delay property. Though, theoretical studies have mainly focused on analyzing throughput of algorithms, delay or queue-size is a very important metric. In practice, routers have finite buffers. Hence, it is possible that among two algorithms, an algorithm with theoretically higher throughput (when buffer-size is infinite) may provide lower throughput compared to the other algorithm in the presence of finite buffer! For example, see Figure 3.1 of Chapter 3 and compare performance of stable Algo2 with un-stable iSLIP algorithm at load $\rho = 0.5$ and buffer-size of 1000.

This motivated us to study the following question: *what is a delay optimal algorithm?*, and *can we compare performance of algorithms in terms of delay?* Traditional methods were not useful in answering these questions. We developed a new approach based on heavy traffic theory to obtain characterization of a delay optimal algorithm. We found that the folklore of Maximum Weight Matching being optimal is false. Further, our results provided an explanation to an intriguing empirical observation made by *Keslassy and McKeown* [2001b] about monotonicity in the delay property of MWM-α algorithms. Separately, our results on heavy traffic analysis of switches are of interest in their own right.

5.1 Future Work

This thesis brings us to a point from which we can follow two seemingly different paths: (i) Implementation of algorithms in actual switches, and (ii) Use and further development of analytic methods of this thesis.

5.1.1 Implementation

Algorithms described in these thesis are very good in performance, verified using theory and via simulations. The claim, which we made repeatedly in this thesis, that still remains

to be verified is about their implementability. We briefly discuss possible application of these algorithms.

The main feature of APSARA algorithm is the possibility of parallel implementation. It is very well suited for switches with very large number of ports, since in such a situation designing centralized scheduler is almost impossible. A possible application of this algorithm can be scheduling in switches for large storage-area networks.

Among SERENA and LAURA algorithms, due to simplicity, we recommend SERENA for the purpose of implementation. For simple implementation of Merge procedure, some form of centralized co-ordination is necessary. This makes SERENA particularly well suited for very high speed switches with fewer ports.

5.1.2 Analytic Method

Though this thesis discusses the design and analysis methods in the context of switch scheduling, we believe that they are quite general.

For example, the heavy traffic analysis of scheduling algorithm for IQ switch should be applicable to a large class of scheduling problems, including the setup of Radio hop introduced by *Tassiulas and Ephremides* [1992]. The use of State Space Collapse for characterizing delay optimal algorithm in the context of switch scheduling is based on a general philosophy. We strongly believe that it should be useful in many other contexts.

The next natural question is: can we use the state space collapse characterization of switches to obtain an estimation of queue-size distribution?

The design methods of the thesis are quite general. For example, the idea of using arrival information in algorithm SERENA can be interpreted by a computer scientist working on online algorithms as "track the adversary." This idea can prove to be very powerful in the context applications like networking where system state changes very slowly.

Bibliography

Anderson, T., S. Owicki, J. Saxe, and C. Thacker, High speed switch scheduling for local area networks, *ACM Transactions on Computer Systems*, *11*, 319–351, 1993. (Cited on pages 7, 15 and 64.)

Asmussen, S., *Applied Probability and Queues*, New York: Wiley, 1987. (Cited on page 25.)

Bertsekas, D. P., A. Nedic, and A. E. Ozdaglar, *Convex Analysis and Optimization*, Athena Scientific, 2003. (Cited on page xvii.)

Billingsley, P., *Probability and Measure*, 3 ed., John Wiley and Sons, 1995. (Cited on page xvii.)

Boyd, S., and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004. (Cited on page xvii.)

Bramson, M., State space collapse with application to heavy traffic limits for multiclass queueing networks, *Queueing Systems*, *30*, 89–148, 1998. (Cited on pages 20, 97, 115 and 116.)

Chang, C., D. S. Lee, and Y. Jou, Load balanced Birkhoff-von Neumann switches, in *IEEE Workshop on High Performance Switching and Routing*, pp. 276–280, 2001. (Cited on page 15.)

Chuang, S.-T., A. Goel, N. McKeown, and B. Prabhakar, Matching output queueing with a combined input output queued switch, *17*, 1030–1039, 1999. (Cited on pages 6, 8 and 15.)

Cisco, Cisco Gigabit Switch Router, 2000, product Overview. (Cited on pages 2 and 19.)

Cormen, T., C. E. Leiserson, and R. L. Rivest, *Inroduction to Algorithms*, MIT Press and McGraw Hill, New York, 1990. (Cited on page xvii.)

Dai, J., and B. Prabhakar, The throughput of switches with and without speed-up, in *Proceedings of IEEE Infocom*, pp. 556–564, 2000. (Cited on pages 16 and 115.)

Dai, J. G., Stability of fluid and stochastic processing networks, *Miscellanea Publication*, 1999. (Cited on page 115.)

Dantzig, G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963. (Cited on page 34.)

Dembo, A., and O. Zeitouni, *Large Deviations Techniques and Applications*, 2 ed., Springer, 1998. (Cited on page 106.)

Durrett, R., *Probability: Theory and Examples*, 2 ed., Duxbury Press, 1995. (Cited on pages xvii and 104.)

Edmonds, J., Maximum Matching and a polyhedron with 0,1-vertices, *Journal of Research of the National Bureau of Standards*, *B 69*, 125–130, 1965. (Cited on page 34.)

Edmonds, J., and R. M. Karp, Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *Journal of ACM*, *18*, 264–284, 1972. (Cited on pages 15 and 34.)

Ganesh, A., N. O'Connell, and D. J. Wischik, *Big Queues*, Springer, 2004, lecture Notes in Mathematics. (Cited on page 106.)

Giaccone, P., D. Shah, and B. Prabhakar, An implementable parallel scheduler for input-queued switches, in *Hot Interconnects 9*, pp. 9–14, 2001. (Cited on pages xviii and 64.)

Giaccone, P., B. Prabhakar, and D. Shah, Towards Simple, High-Performance Schedulers for High-aggregate bandwidth Switches, 2002. (Cited on pages xviii and 64.)

Giaccone, P., B. Prabhakar, and D. Shah, Randomized scheduling algorithms for high-aggregate bandwidth switches, *IEEE Journal on Selected Areas in Communications High-performance electronic switches/routers for high-speed internet*, *21*, 546–559, 2003. (Cited on pages xviii, 16 and 64.)

Hajek, B., Hitting and occupation time bounds implied by drift analysis with applications, *Advances of Applied Probability*, *14*, 502–525, 1982. (Cited on page 35.)

Harrison, J. M., *Brownian Motion and Stochastic Flow Systems*, John Wiley and Sons, New York, 1985. (Cited on page 116.)

Harrison, J. M., Brownian models of queueing networks with heterogeneous customer populations, *Stochastic Differential Systems, Stochastic Control Theory and Application*, pp. 147–186, 1988, iMA Volumes in Mathematics and Its Applications. (Cited on page 116.)

Harrison, J. M., Balanced fluid models of multiclass queueing networks: A heavy traffic conjecture, *Sotchastic Networks*, *71*, 1–20, 1995, iMA Volumes in Mathematics and Its Applications. (Cited on page 116.)

Harrison, J. M., and R. J. Williams, Brownian models of feedforward queueing networks: Quasireversibility and product form solutions, *Annals of Applied Probability*, *2*, 263–293, 1992. (Cited on page 116.)

H.Duan, J. W. Lockwood, S. M. Kang, and J. D. Will, A high performance OC12/OC48 queue design prototype for input buffered ATM switches, in *IEEE INFOCOM*, vol. 1, pp. 20–28, 1997. (Cited on pages 15 and 64.)

Hennesy, P., and D. Patterson, *Computer Architecture: A Quantitive Approach*, Morgan Kauffman, 1986. (Cited on pages xvii and 5.)

Iglehart, D. L., and W. Whitt, Multiple channel queues in heavy traffic I, *Advances in Applied Probability*, *2*, 150–177, 1970a. (Cited on page 116.)

Iglehart, D. L., and W. Whitt, Multiple channel queues in heavy traffic II, *Advances in Applied Probability*, *2*, 355–364, 1970b. (Cited on page 116.)

Iglehart, D. L., and W. Whitt, The equivalence of functional cnetral limit theorems for counting processes and associated partial sums, *Annals of Mathemical Statistics*, *42*, 1372–1378, 1971. (Cited on page 116.)

Iyer, S., The Parallel Packet Switch Architecture, Ph.D. thesis, 2002. (Cited on pages 6 and 15.)

Iyer, S., R. Zhang, and N. McKeown, Routers with a Single Stage of Buffering, in *ACM SIGCOMM*, 2002. (Cited on page 6.)

Karatzas, I., and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2 ed., Springer, 1991. (Cited on page xvii.)

Karol, M., M. Hluchyj, and S. Morgan, Input Versus Output Queueing on a Space Division Packet Switch, *IEEE Transactions on Communications*, *35*, 1347–1356, 1987. (Cited on page 7.)

Karol, M., K. Eng, and H. Obara, Improving the Performance of Input-Queued ATM Packet Switch, in *IEEE INFOCOM*, pp. 110–115, 1992. (Cited on page 7.)

Keshav, S., and R. Sharma, Issues and trends in router design, *36*, 144–151, 1998. (Cited on page xvii.)

Keslassy, I., The Load-Balanced Router, Ph.D. thesis, 2004, phD Thesis, Department of Electrical Engineering, Stanford University. (Cited on page 6.)

Keslassy, I., and N. McKeown, Analysis of Scheduling Algorithms That Provide 100% Throughput in Input-Queued Switches, in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001a. (Cited on pages 16, 20 and 115.)

Keslassy, I., and N. McKeown, Analysis of scheduling algorithms that provide 100% throughput in Input-Queued switches, in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001b. (Cited on page 118.)

Krishna, P., N. S. Patel, A. Charny, and R. J. Simcoe, On the speedup required for work-conserving crossbar switches, in *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1057–1066, 1999. (Cited on pages 6 and 15.)

Kumar, P. R., and S. P. Meyn, Stability of Queueing Networks and Scheduling Policies, *IEEE Transactions on Automatic Control*, *40*, 251–260, 1995. (Cited on page 25.)

Kumar, S., and P. R. Kumar, Performance Bounds for Queueing Networks and Scheduling Policies, *IEEE Transactions on Automatic Control*, *38*, 1600–1611, 1994. (Cited on page 35.)

Leonardi, E., M. Mellia, F. Neri, and M. A. Marsan, Bounds on Average Delays and Queue Size Averages and Variances in Input Queued Cell-Based Switches, in *IEEE INFOCOM*, pp. 1095 –1103, 2001. (Cited on pages 17 and 35.)

Marsan, M. A., A. Bianco, E. Leonardi, and L. Milia, RPA: a flexible scheduling algorithm for input buffered switches, *IEEE Transaction on Communications*, *47*, 1921–1933, 1999. (Cited on pages 15 and 64.)

Marsan, M. A., P. Giaccone, E. Leonardi, and F. Neri, On the stability of local scheduling policies in networks of packet switches with input queues, *IEEE Journal on Selected Areas in Communications "High-performance electronic switches/routers for high-speed internet"*, *21*, 642–655, 2003. (Cited on page 16.)

McKeown, N., Growth in Router Capacity, *Personal Communication and Talks*. (Cited on page 5.)

McKeown, N., Scheduling Algorithms for Input-Queued Cell Switches, Ph.D. thesis, 1995, phD Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley. (Cited on pages 15 and 64.)

McKeown, N., iSLIP: a scheduling algorithm for input-queued switches, *IEEE Transaction on Networking*, *7*, 188–201, 1999. (Cited on pages 15 and 64.)

McKeown, N., V. Anantharam, and J. Walrand, Achieving 100% throughput in an input-queued switch, in *Proceedings of IEEE Infocom*, pp. 296–302, 1996. (Cited on pages 14, 16 and 34.)

Mekkittikul, A., and N. McKeown, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, in *IEEE INFOCOM*, pp. 792–799, 1998. (Cited on pages 105 and 115.)

Meyn, S. P., and R. L. Tweedie, *Markov Chains and Stochastic Stability*, Springer-Verlag, London, 1993a. (Cited on page 25.)

Meyn, S. P., and R. L. Tweedie, Stability of Markovian Processes II: Continuous time processes and sampled chains, *Advances of Applied Probability*, *25*, 487–517, 1993b. (Cited on page 25.)

Meyn, S. P., and R. L. Tweedie, stability of Markovian Processes III: Foster-Lyapunov criteria for Continuous time processes, *Advances of Applied Probability*, *25*, 518–548, 1993c. (Cited on page 25.)

Motwani, R., and P. Raghavan, *Randomized algorithms*, Cambridge University Press, 1995. (Cited on pages xvii and 18.)

Munkres, J., *Topology*, 3 ed., Prentice Hall, 1999. (Cited on page xvii.)

Nijenhuis, A., and H. Wilf, *Combinatorial algorithms: for computers and calculators*, 2 ed., Academic Press, 1978. (Cited on pages xvii and 41.)

Prabhakar, B., and N. McKeown, On the speedup required for combined input and output queued switching, in *Automatica*, pp. 1909–1920, 1999. (Cited on pages 6, 8, 15 and 17.)

Reimann, M. I., Open queueing networks in heavy traffic, *Math. Oper. Res.*, *9*, 441–458, 1984. (Cited on page 116.)

Shah, D., Stable algorithms for Input Queued Switches, in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001. (Cited on pages xviii, 16 and 115.)

Shah, D., Maximal Matching Scheduling is good enough, in *Proceedings of IEEE Globecom*, 2003. (Cited on pages 6 and 35.)

Shah, D., and M. Kopikare, Delay bounds for the approximate Maximum Weight matching algorithm for input queued switches, in *Proceedings of IEEE Infocom*, 2002. (Cited on pages xviii and 35.)

Shah, D., and D. J. Wischik, An Optimal Scheduling Algorithm for Input Queued Switch under Heavy Traffic, *Preprint*. (Cited on pages xviii and 115.)

Shah, D., P. Giaccone, and B. Prabhakar, An efficient randomized algorithm for input-queued switch scheduling, in *Hot Interconnects 9*, pp. 3–8, 2001. (Cited on pages xviii and 64.)

Simmons, G. F., *Introduction to Topology and Modern Analysis*, McGraw-Hill text, 1963. (Cited on page xvii.)

Sleator, D. D., and R. E. Tarjan, Amortized efficiency of list update and paging rules, *Communications of the ACM*, *28*, 202–208, 1985. (Cited on page 35.)

Stanley, R. P., *Enumerative Combinatorics*, vol. 2, Cambridge University Press, 1999. (Cited on page xvii.)

Stolyar, A. L., MaxWeight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic, *Annals of Applied Probability*, *14*, 1–53, 2004. (Cited on pages 19, 20 and 115.)

Tamir, Y., and H. Chi, Symmetric crossbar arbiters for VLSI communication switches, *IEEE Transaction on Parallel and Distributed Systems*, *4*, 13–27, 1993. (Cited on pages 7, 15 and 64.)

Tarjan, R. E., *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983. (Cited on pages xvii and 34.)

Tassiulas, L., Linear complexity algorithms for maximum throughput in radio networks and input queued switches, in *IEEE INFOCOM*, vol. 2, pp. 533–539, 1998. (Cited on pages 2, 16 and 39.)

Tassiulas, L., and A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, *IEEE Transactions on Automatic Control*, *37*, 1936–1948, 1992. (Cited on pages 14, 16, 34, 64, 115 and 119.)

van Lint, J. H., and R. M. Wilson, *A course in Combinatorics*, 1 ed., Cambridge University Press, 1992. (Cited on page xvii.)

West, D. B., *Introduction to Graph Theory*, Prentice Hall, 1996. (Cited on page xvii.)

Whitt, W., Weak convergence theorems for priority queues: Reemptive resume discipline, *Journal of Applied Probability*, *8*, 74–94, 1971. (Cited on page 116.)

Williams, R., Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse, *Queueing Systems*, *30*, 27–88, 1998. (Cited on pages 20, 115 and 116.)