High Performance Queueing and Scheduling in Support of Multicasting in Input-Queued Switches

Weiying Zhu
Old Dominion University

HIGH PERFORMANCE QUEUEING AND SCHEDULING IN SUPPORT OF MULTICASTING IN INPUT-QUEUED SWITCHES

by

Weiying Zhu
B.S. July 1996, Xi'an Jiaotong University
M.S. June 1999, Huazhong University of Science and Technology

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
August 2006

Approved by:

Min Song (Director)

James F. Leathrum, Jr. (Member)

K. Vijayan Asari (Member)

Hussein Abdel-Wahab (Member)

ABSTRACT

HIGH PERFORMANCE QUEUEING AND SCHEDULING IN SUPPORT OF MULTICASTING IN INPUT-QUEUED SWITCHES

Weiying Zhu
Old Dominion University, 2006
Director: Dr. Min Song

Due to its mild requirement on the bandwidth of switching fabric and internal memory, the input-queued architecture is a practical solution for today's very high-speed switches. One of the notoriously difficult problems in the design of input-queued switches with very high link rates is the high performance queueing and scheduling of multicast traffic. This dissertation focuses on proposing novel solutions for this problem. The design challenge stems from the nature of multicast traffic, i.e., a multicast packet typically has multiple destinations. On the one hand, this nature makes queueing and scheduling of multicast traffic much more difficult than that of unicast traffic. For example, virtual output queueing is widely used to completely avoid the head-of-line blocking and achieve 100% throughput for unicast traffic. Nevertheless, the exhaustive multicast virtual output queueing is impractical and results in out-of-order delivery. On the other hand, in spite of extensive studies in the context of either pure unicast traffic or pure multicast traffic, the results from a study in one context are not applicable to the other context due to the difference between the natures of unicast and multicast traffic. The design of integrated scheduling for both types of traffic remains an open issue.

The main contribution of this dissertation is twofold: firstly, the performance of an interesting approach to efficiently mitigate head-of-line blocking for multicast traffic is theoretically analyzed; secondly, two novel algorithms are proposed to efficiently integrate unicast and multicast scheduling within one switching fabric.

The research work presented in this dissertation concludes that (1) a small number of queues are sufficient to maximize the saturation throughput and delay performances of a large multicast switch with multiple first-in-first-out queues per input port; (2) the theoretical analysis results are indeed valid for practical large-sized switches; (3) for a large $M \times N$ multicast switch, the final achievable saturation throughput decreases as the ratio of M/N decreases; (4) and the two proposed integration algorithms exhibit promising performances in terms of saturation throughput, delay, and packet loss ratio under both uniform Bernoulli and uniform bursty traffic.

To my parents,

Yisheng and Suzhen,

my husband, Chuan,

and my daughter, Catherine.

ACKNOWLEDGMENTS

I would like to express my gratitude to the many people who helped make this dissertation possible. In particular, I would like to thank:

– my advisor, Dr. Min Song, for his guidance, enthusiasm and unfailing support,

– the other members of my dissertation committee, Dr. Hussein Abdel-Wahab, Dr. K. Vijayan Asari, and Dr. James Leathrum, for many helpful comments on this dissertation and full support to my research work,

– Dr. Steven Gray for his invaluable instructions and suggestions on both mathematics and career,

– current and previous members of Networking Research Lab for many stimulating discussions and their friendship, and

– all my friends and colleagues in the department of Electrical and Computer Engineering who made my time here become such a good memory.

Last but not the least, I would like to thank my parents, my brother, and my husband for their love and support throughout all these years, and my little daughter for the great happiness brought by her charming smile and laugh.

I would also like to acknowledge the financial support I received from the department of Electrical and Computer Engineering and Old Dominion University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

One of the notoriously difficult problems for today's switches and routers with very high link rates is to support multicasting with high performance. This dissertation focuses on proposing practical solutions on this topic. System performance is evaluated through both theoretical analysis and simulation study. This chapter introduces the background, motivation, research goals and outline of this dissertation.

This chapter is organized as follows. Section 1.1 discusses the background and motivation of the research work. Section 1.2 defines three common metrics (saturation throughput, delay, and packet loss ratio) that are used to evaluate the performance of an input-queued packet switch. Section 1.3 states the main goals that are going to be achieved. Finally, the outline of this dissertation is given in Section 1.4.

1.1 Background and Motivation

As a platform for conducting research, education, and business, the Internet needs to provide efficient support for the rapidly increasing multicast applications such as distributed interactive simulations, distance learning, digital video libraries, video-on-demand, and video conference. Since the support of multicasting is notoriously difficult [8], in today's Internet, the dominant model of communication is "unicast", i.e., the sender must create a separate copy of the data for each recipient. A major advantage of using multicasting is the decrease of network load, especially for the applications with many recipients and a large amount of data, e.g., streaming video. Possessing a very fast input/output link rate is another character of switches and routers in today's Internet. For example, Cisco 12000 GSR [31], MGR [42], both of which are 50-Gb/s IP routers, and Tiny Tera [33], which is a 0.5-Tb/s MPLS switch, have been designed. As a result, there

In this dissertation, *IEEE/ACM Transactions on Networking* is used as the journal model for formatting.

is an increasing demand for high speed switches and routers that support multicasting with high performance.

Most multicast applications are also multimedia streaming applications such as live video broadcasts, distance learning, and corporate telecasts. Web sites today offer streaming audio and video of news broadcasts, music television, live sporting events and more [10, 23]. Pulled by market demand and pushed by technology, multimedia streaming traffic increases dramatically [7]. At the beginning of the 21st century, audio and video streaming traffic has already become an important ingredient of Internet traffic. Plonka found that 23% of the traffic at the University of Wisconsin-Madison was due to digital audio in March 2000 [43]. A 1999 study by Wolman et al. showed that 18-24% of Web-related traffic entering the University of Washington was multimedia streaming traffic [52]. An industry study in September 2000 [11] showed that 60 million people listened to or watched streaming media each month, 58 US TV stations performed live web-casting, 34 programs offered on-demand streaming media, and 69 international TV web-casters existed [37]. On the other hand, it is observed that live streams have diverse clients. A live stream generally spans hundreds of AS domains and tens of countries. A 2004 study [48] showed that most of the streams reached 11 or more different time zones, 10 or more different countries, and 100 or more different AS domains. Due to the large number of multicast applications and the diversity of clients accessing live streams, the incoming traffic at an input port of a switch is a combination of multicast flows coming from a large number of different sources and spanning more different destinations. Consequently the assumption that the traffic is uniformly distributed among input ports and output ports is reasonable and does not lose generalization. Uniform traffic is widely used to evaluate multicast switch performance [4, 6, 14, 40, 44].

Before further discussion, the difference between the natures of unicast and multicast packets needs to be clarified: one unicast *input* packet generates one *output* packet; one multicast *input* packet with a fanout size of F generates F *output* packets. The input load is defined as the number of input packets arriving in a time slot at an input port. The output load is defined as the number of output packets arriving in a time slot with a destination to an output port.

From the switch architecture point of view, input-queued architectures have received considerable research attention [9, 30, 49]. This is due to the fact that the speed of switching fabric and internal memory of an input-queued switch is only required to be the same as the link rate, which is very promising for large-sized routers and switches with extremely fast link rates. Instead, for an $N \times N$ output-queued unicast switch, the speed of switching fabric and output buffer is required to be up to N times the link rate; for an $N \times N$ output-queued multicast switch, the speed of switching fabric and output buffer is required to be up to N^2 times the link rate. This is not practical for switches and routers in very high-speed networks [20]. Therefore, this dissertation focuses on input-queued architectures.

The switching fabric, queueing policy, and scheduling algorithm are three essential components of input-queued switches. In the past decade, a number of switching fabric architectures have been proposed and deployed to support multicasting [3, 16, 17, 18, 24, 27, 28, 45, 53]. Assuming that a multicast switch fabric with a speedup of one has been well-designed, this dissertation focuses on the study of high performance queueing and scheduling of multicast traffic. Within a multicast switch fabric with a speedup of one, in each time slot, no more than one input packet can be sent from an input port; and no more than one output packet can be delivered to an output port; yet multiple output packets belonging to the same input packet can be delivered to the corresponding output ports.

An assumption in this dissertation is that the packets transmitted by the switching fabric are assumed to have equal length. The time slot coincides with the time needed to transfer one packet across the switching fabric. The incoming variable-sized packets are segmented into fixed-sized packets (cells) before entering input queues and segments are put back together before departing from output ports. This assumption does not lose generality because almost all practical IP switches and routers currently use a cell-based fabric [1, 8, 30, 31, 33, 42, 44, 46, 47, 49].

1.2 Performance Metrics for Input-Queued Switches

Generally speaking, saturation throughput, delay, and packet loss ratio are three common metrics used to evaluate the performance of an input-queued packet switch. Their definitions are clarified as follows.

In each time slot, for every output packet scheduled to be transmitted, a match is set up between the input port where it arrives and the output port where it is destined after performing the scheduling algorithm. Throughput is defined as the number of matches over the number of output ports in a time slot. For a given switch, there is a threshold value of throughput that is called saturation throughput. Assuming that the input buffer is big enough to avoid packet loss, throughput equals to the output load in a switch with a speedup of one when the output load is less than the saturation throughput. Once the output load exceeds the saturation throughput, the switch is saturated and its throughput equals to the saturation throughput no matter how much the output load is increased. Saturation throughput indicates the switching capacity of a given switch. If the saturation throughput of a switch with a given scheduling algorithm equals to one, which is the maximum value due to the speedup of one, it is said that the given scheduling algorithm can achieve 100% throughput.

The second performance measurement, delay, is defined as the number of time slots spent by an output packet in the switching system, i.e., the difference between the moment when an output packet is delivered to the output link and the moment when its corresponding input packet enters the input queue.

The third performance measurement, packet loss ratio, is defined as the number of dropped output packets over the total number of output packets in a certain number of time slots.

For a switch with a scheduling algorithm that can achieve 100% throughput, there exists a finite input buffer size such that no incoming packet will be dropped, no matter how heavy the output load is. The switch is guaranteed to be stable under any admissible traffic. If the scheduling algorithm cannot achieve 100% throughput, once the output load exceeds the saturation throughput, no matter how big the input buffer size is, some incoming packets will be dropped. In other words, no matter how big the input buffer is, the scheduling algorithm can only potentially work within a subset of the whole admissible interval of output load. Therefore, saturation throughput is the first and most important measurement used to evaluate scheduling algorithms. Compared with a scheduling algorithm without 100% throughput, a scheduling algorithm with 100% throughput is preferred. However, saturation throughput is not the only standard.

Comparing two scheduling algorithms, both of which can achieve 100% throughput, the one with smaller delay and smaller packet loss ratio is preferred.

1.3 Problem Statement

The main goals of this dissertation are as follows:

1. To theoretically analyze the saturation throughput and delay performances of multicast switches with multiple queues per input port. The multicast traffic is evenly distributed among all input ports. The analysis model is built for $N \times N$ switches.
2. To generalize the analysis in goal 1 to the case that the multicast traffic is gathered among fewer input ports and engages more output ports. The analysis model is built for $M \times N$ switches.
3. To validate the theoretical results achieved in goals 1 and 2 using extensive experimental data obtained through simulations.
4. To propose novel integration algorithms to integrate unicast and multicast scheduling within one switching fabric. The promising performance of the proposed algorithms is demonstrated through both analysis and simulations.

1.4 Dissertation Outline

The remaining part of this dissertation is organized as follows. Chapter II reviews the recent literature on state-of-the-art queueing and scheduling techniques for input-queued switches and identifies the open issues on the support of multicasting in input-queued switches. In Chapter III the closed-form expressions of saturation throughput, average service time, and average delay are theoretically derived for large multicast switches with multiple queues per input port. The incoming multicast traffic follows Poisson distribution and is evenly distributed among all input ports. Random queueing and scheduling policies are adopted. Extensive simulations are also performed to validate the theoretical analysis and infer further conclusions. Chapter IV generalizes both the theoretical analysis and the simulation study to the case that the incoming multicast traffic gathers among fewer input ports and engages more output ports. The integrated unicast and multicast scheduling within one switching fabric is presented in Chapter V.

It is shown that the proposed integration algorithms exhibit a promising performance. Finally, concluding remarks and future research are given in Chapter VI.

CHAPTER II

RELATED WORK

This chapter reviews the recent literature on state-of-the-art queueing and scheduling techniques for input-queued switches. Instead of attempting to give a comprehensive survey of the field, the review focuses on the techniques related to the work presented in this dissertation. The open issues on queueing and scheduling in support of multicasting in input-queued switches are identified.

This chapter is organized as follows. Section 2.1 examines the virtual output queueing (VOQ) [1, 50] technique and the suitable schedulers such as PIM [1], LQF [34], OCF [34], LPF [36], and *i*SLIP [32], which are designed to completely avoid head-of-line (HOL) blocking and achieve 100% throughput for unicast traffic. Queueing policies and scheduling algorithms to mitigate HOL blocking for multicast traffic in input-queued switches are discussed in Section 2.2. Section 2.3 summarizes relevant work in integrating unicast and multicast scheduling within one switching fabric and points out the weakness of current solutions.

2.1 VOQ and Schedulers for VOQ-Based Unicast Switches

Two critical components in an input-queued switch, queueing and scheduling, have been extensively studied for unicast traffic. It is well known that HOL blocking limits the saturation throughput of an input-queued switch with single first in first out (FIFO) queue per input port. In each time slot, at an input port, only the packet at the HOL position, called the HOL packet, is eligible for being transmitted. When the HOL packet is blocked, all the packets behind it in the queue are prevented from being transmitted even if the output port they need is idle. Even with benign traffic, saturation throughput is limited by HOL blocking to just $(2-\sqrt{2}) \approx 58.6\%$ [19].

To completely avoid HOL blocking, VOQ is first designed in [50] by Tamir *et al.* for an $N \times N$ input-queued unicast switch. At each input port, the input buffer is organized as

N FIFO queues with each corresponding to an output port. The incoming packets with the same destined output port are buffered into the corresponding queue according to its destined output port. All the HOL packets are eligible for transmission in a time slot. In [1], adopting VOQ, an algorithm, named parallel iterative matching (PIM), is proposed by Anderson *et al.* to find a maximal matching between input and output ports by using parallelism, randomness, and iteration. Maximal matching [1, 39] is a matching in which no unmatched input port has a queued packet destined for an unmatched output port. PIM iterates three steps (Request, Grant, and Accept) until a maximal matching is found or a fixed number of iterations are performed: each unmatched input port sends a request to every output port for which it has a queued packet; if an unmatched output receives any requests, it randomly grants one of the requests; if an input receives any grants, it randomly accepts one of the grants and is matched with the output port who issues that grant. Through simulations, the throughput and delay performances are evaluated. It is shown that PIM can achieve a nearly ideal match in an average of $O(\log N)$ iterations. Moreover, the hardware requirements are modest enough to make VOQ and PIM practical for high-speed switching. Due to its promising performance, VOQ attracts researchers' attention and is widely adopted in the design of unicast switches. A number of further research efforts have been made for VOQ-based switches.

In [39], Nong *et al.* theoretically analyze the input-queued unicast switch using VOQ as the queueing policy and PIM as the scheduling algorithm. A closed-form solution for saturation throughput as a function of switch size and number of iterations is derived. It is found that four iterations are sufficient for achieving a saturation throughput of about 99% for a switch of any size. Using the tagged input queue approach, an analytical model is developed for switches under an independent identically distributed Bernoulli traffic, whose destinations are uniformly distributed among output ports. Throughput, average packet delay, and packet loss ratio are computed from the analytical model. The study given in [39] provides the theoretical support to the conclusions that HOL blocking is completely avoided by the VOQ technique and 100% throughput can be achieved by PIM.

The theoretical performance analysis on unicast switches with a certain number m of FIFO queues per input port and a speedup of one is given in [22] by Kim *et al.* The output ports are participated into *m* groups. At an input port, each of the *m* queues is dedicated to buffering packets destined to a particular group of output ports. The scheduling algorithm is essentially based on PIM. Assuming that the switch size is very large, Kim *et al.* derive the closed formulas for throughput bound, average packet delay, average queue length, and packet loss bound as the function of *m*. The theoretical numerical results prove that the HOL blocking probability decreases as *m* increases. When *m* equals to the number of output ports, the switch discussed in [22] is a VOQ-based switch and the probability that HOL blocking occurs becomes zero.

It is proved that maximum weight matching scheduling algorithms can provide the best performance for VOQ-based switches [34, 51]. In [26], Leonardi *et al.* theoretically derive upper bounds on average delay, average queue length, and variance of queue length for unicast input-queued switches that adopt VOQ for queueing and maximum weight matching [51] for scheduling. Two maximum weight matching algorithms, longest queue first (LQF) and oldest cell first (OCF), are proposed in [34]. It is concluded that both LQF and OCF can lead to a saturation throughput of 100% for independent and either uniform or non-uniform traffic. However, the complexity of the most efficient maximum weight matching algorithms is $O(N^3 \log N)$. This is too complex to be practical for high speed switches. On the contrary, another class of algorithms, named maximum size matching algorithms [34, 51], attempt to maximize the number of matches between input and output ports in each time slot. Generally speaking, maximum size matching algorithms are simpler than maximum weight matching algorithms and perform well when the incoming traffic is uniformly distributed among output ports. Unfortunately, maximum size matching algorithms perform poorly when the incoming traffic is non-uniform. Longest port first (LPF) is designed in [36] to take the advantage of both maximum size matching and maximum weight matching. The complexity of LPF is $O(N^{2.5})$, lower than LQF and OCF. Meanwhile, LPF can also achieve 100% throughput for both uniform and non-uniform traffic.

An iterative round-robin algorithm, *i*SLIP, is studied in [32]. As a high throughput, starvation free, fast, and simple to implement in hardware algorithm, together with VOQ, *i*SLIP is a practical unicast scheduling solution for high-performance switches and routers. *i*SLIP can achieve 100% throughput for uniform traffic. For non-uniform traffic, *i*SLIP adapts to a fair scheduling policy that never starves an input queue. Due to the simplicity of hardware implementation, the scheduler of a 32-port switch can be built on single chip and make approximately 100 million scheduling decisions per second. More details of *i*SLIP are given in Chapter V.

2.2 Mitigating HOL Blocking for Multicast Traffic in Input-Queued Switches

The design challenge of multicast scheduling and queueing stems from its nature, i.e., a multicast packet typically has multiple destinations. The vector of destinations is named as *fanout set*. The number of destinations is named as *fanout size*. Only after every destination output port receives the respective output packet generated by the input multicast packet, can that input packet be removed from the input queue. There are two kinds of service disciplines: *fanout splitting* and *no fanout splitting*. With no fanout splitting, all the output packets of an input multicast packet must be sent to the corresponding output ports in one time slot. With fanout splitting, a multicast packet could be delivered to its destination output ports in more than one time slots and maybe only partial destinations are served. At the end of a time slot, for a multicast packet, the vector of the unserved destinations is denoted as *residue* and the number of the unserved destinations is named as *residue size*.

In the design of multicast switches, it is straightforward to allocate single FIFO queue at each input port, which has been studied in different context. In [5], two input access mechanisms, cyclic priority reservation and neural-network-based access, are compared for the multicast switch with a service discipline of no fanout splitting. The saturation throughput and delay performances are evaluated through simulations. Under the assumptions of no fanout splitting and random packet selection policy, the throughput, delay, and packet loss probability performances of large-sized multicast switches are theoretically analyzed in [35]. The delay performance of small-sized multicast switches is theoretically studied in [13]. In [14], Hui *et al.* theoretically analyzes the performance of large-sized multicast switches in terms of saturation throughput and average waiting time at the HOL position. The analysis in both [13] and [14] is under the assumptions of fanout splitting, random selection policy for settling output port contention, and the traffic that is uniformly distributed among input ports and output ports. Three fanout splitting multicast scheduling algorithms are presented in [44], the Concentrate algorithm, TATRA, and WBA. Their performances are studied through simulations. The Concentrate algorithm always concentrates the residue onto as few inputs as possible, which leads to high throughput and low delay. However, it can starve some input ports indefinitely. TATRA avoids starvation by using a strict definition of fairness, while comparing well to the Concentrate algorithm. Both the Concentrate algorithm and TATRA are difficult to be implemented using hardware. In WBA, weights are assigned to HOL packets based on their age and residue size. While more than one HOL packets contend for an output port, the one with the heaviest weight is selected. WBA is very simple to be implemented using hardware and allows the designer to balance the tradeoff between fairness and throughput. All the above studies demonstrate that the saturation throughput of multicast switches with single FIFO queue per input is compromised by the HOL blocking.

Multiple slot cell scheduling algorithms are proposed to mitigate HOL blocking [6, 21]. Packets behind the HOL packet are allowed to be transmitted prior to the HOL one. Based on simulations, it is shown that the increased scheduling space results in the increase of both throughput and delay performances. However, two major prices are paid for this increment. One is that the packet delivery is out-of-order; the other is that input queues need to have random access capability, which is much more complex than FIFO queues.

In [40], Pan *et al* propose a multicast queueing scheme for an $N \times N$ switch by utilizing the VOQ structure for unicast traffic, which is similar to the one proposed in [38]. A multicast packet with a fanout size of F is separately stored as $(F + 1)$ packets: one data packet and F address packets. At each input port, a shared data queue buffering data packets and N VOQs buffering address packets are designed. This scheme can avoid HOL blocking and achieve 100% throughput for uniform traffic together with a suitable scheduling algorithm. However, this scheme requires that the writing bandwidth of input buffer is up to $(N + 1)$ times the bandwidth of input link. In addition, the input buffer needs to support random access. Hence, the hardware implementation of this scheme is much more expensive than that of pure FIFO queueing policies.

Marsan *et al.* discuss the maximum throughput of an $N \times N$ multicast switch with *exhaustive* multicast virtual output queueing (MC-VOQ) [30]. At each input port, (2^N – 1) FIFO queues are provided with each for a possible fanout set. A partially served HOL packet is re-enqueued into another FIFO queue according to its residue. Although this scheme can completely avoid HOL blocking, it leads to out-of-order delivery of packets and is not practical since the number of queues per input port increases exponentially as the switch size increases.

A queueing scheme, called per-multicast-flow queueing, is proposed in [29] to mitigate HOL blocking. At an input port, a FIFO queue is allocated for each multicast flow. The complexity of the scheduling algorithm depends on the maximum number of queued multicast flows and the performance highly depends on traffic patterns. Thus, it is not practical for the traffic that contains a number of multicast flows.

One interesting approach to alleviate HOL blocking is to allocate a certain number (K) of FIFO queues per input port. Although all the HOL packets in an input port's local queues are eligible for transmission, only one of them will be selected. This queueing scheme can assure in-order delivery of packets and is practical to be implemented. Simulation studies are given in [12] and [4]. Gupta *et al.* experiment with two queueing schemes, Split and Majority, and two scheduling algorithms, MaxWeight and MaxService, in [12]. The set of output ports is partitioned into K subsets with each being represented by a bit-mask and belonging to one of the K queues at an input port. In the Split scheme, if the fanout set of an incoming packet completely fits into one queue, it gets into that queue; otherwise, it gets split into multiple queues. In the Majority scheme, the incoming packet is put into the queue whose bit-mask is the most in common with the packet's fanout set. In the MaxWeight algorithm, each input port prioritizes the HOL packet with the highest weight. The weight is assigned according to age and fanout size of the packet. In the MaxService algorithm, each input port prioritizes the HOL packet with the highest discharge percentage. The discharge percentage of a HOL packet is defined as the number of grants for transmission issued by output ports over the residue size of that packet. The simulation results on the average packet delay given in [12] indicate that Majority performs much better than Split and MaxService performs much better than MaxWeight. The difficulty of the Majority scheme is the way to design bit-masks for the queues. In [12], no clear solution is given on this issue. When the number of queues is close to or even bigger than the number of output ports, the assignment of bit-masks does not adequately capture fanout sets of multicast packets. The fanout set of a packet is the same in common with the bit-masks of a number of queues. In [4], three queueing policies and three scheduling algorithms with fanout splitting are discussed. With the Random Queueing (RQ) policy, a multicast flow is associated randomly with one of the *K* queues. With the Minimum-Distance Queueing (MDQ) policy, each queue is associated with a representative fanout set. A multicast flow is allocated to a queue such that the Hamming distance between the representative fanout set of the queue and the fanout set of the multicast flow is minimized. With the Load-Balanced Queueing (LBQ) policy, multicast flows are partitioned into the *K* queues according to their fanout sets such that the output loads for queues are equalized. With the Random Scheduler (RS), both input port contention and output port contention are settled by random selection among the candidates. With the Greedy Scheduler (GS), according to queue length and residue size, a weight is assigned to every HOL packet. Matches are set up between input and output ports in several iterations with priority being given to HOL packets with heavier weights. The Greedy Min-Split Scheduler (GMSS) also matches input and output ports in multiple iterations and gives priority to the HOL packets with larger residue size for service. Saturation throughput is evaluated through simulations for different combinations of scheduler and queueing policy. It is shown that GMSS performs the best among schedulers and LBQ is the best queueing policy. They also conclude that a small number of queues (for example, twice the number of output ports) are sufficient to achieve the highest saturation throughput. However, both GS and GMSS are centralized schedulers, i.e., all HOL packets across all input ports need be examined in sequence in several iterations in order to make the scheduling decision. This is not scalable for large-sized switches. Furthermore, neither [4] nor [12] provides theoretical analysis on multicast switches with multiple FIFO queues per input port to support their conclusions.

2.3 Queueing and Scheduling in Input-Queued Switches for Hybrid Traffic

Both unicast and multicast queueing and scheduling aim to achieve high throughput, small delay, low packet loss ratio, starvation free, etc. However, multicast queueing and scheduling are totally different from unicast queueing and scheduling. An incoming unicast packet only has one destined output port; on the contrary, an incoming multicast packet has more than one destined output ports. Due to the different characteristics of traffic, typically queueing policies and scheduling algorithms are studied and proposed for unicast traffic and multicast traffic separately. Unicast queueing policies and scheduling algorithms could not handle the case that incoming packets have more than one destined output ports. In addition, multicast queueing policies and scheduling algorithms do not perform well for unicast traffic. One approach of integration is to use isolated switching fabrics for unicast and multicast traffic such that different algorithms can be used respectively. This approach does not fully utilize the resource of switching fabrics [2]. The other alternative approach is to coordinate unicast scheduling and multicast scheduling together within one switching fabric so that the switching fabric can be fully utilized.

Andrews *et al.* proposed an integrated scheduling procedure that packs unicast packets into idle slots left by the multicast schedule [2]. They successfully showed that (1) theoretically, both the optimal unicast integration problem and the optimal multicast scheduling problem are NP-hard; and (2) an alternate on-line algorithm for unicast integration can find a match within a factor of α of optimal and achieve 2α-competitive. Their simulation results indicated that the integration procedure is efficient for small multicast rates while the overall throughput (sum of unicast and multicast throughputs) drops significantly for higher multicast rates. When multicast output load is greater than 0.4, the system becomes unstable since multicast queues become unstable. The overall throughput only can achieve approximately 50% when multicast output load approaches to 0.4. This is because only one multicast queue is provided at each input port and the multicast scheduling is not efficient enough to achieve a higher multicast throughput. Another issue of this integration procedure is that unicast traffic is always scheduled with lower priority in each time slot by using the available input/output ports left by multicast scheduling. Hence, fair resource allocation is not guaranteed for unicast traffic and more efficient multicast scheduling may reduce the space available to unicast.

In the multicast split/merge (MSM) algorithm presented by Minkenberg [38], at each input port, a shared memory and N VOQs are designed. An incoming packet is put into the shared memory and its memory address is duplicated to each of its destination VOQs with each copy being scheduled independently using unicast scheduling algorithm. Such a scheme requires that the writing bandwidth of input buffer is twice the bandwidth of input link. In addition, besides FIFO buffers, a memory with random access is needed. By logically splitting multicast packets into unicast packets, multicast scheduling is integrated with unicast scheduling naturally, and 100% throughput can be achieved. Through simulation study, it is shown that in a combined input- and output-queued switch, MSM exhibits a better performance than the Concentrate algorithm [44] designed for an input-queued switch with only one multicast queue per input port. However, in an input-queued switch, MSM's delay performance is even significantly worse than Concentrate at medium multicast output loads.

The integrated scheduling algorithm offered by Lee *et al.* in [25] handles both unicast and multicast traffic concurrently using two buffers at each input port with one for unicast packets and the other for multicast packets. Even with uniform Bernoulli traffic, their simulation reveals that the saturation throughput is less than 80%. This result is not surprising considering that the switch is an input-queued switch with only one unicast FIFO queue and one multicast FIFO queue, because of the HOL blocking.

To our best knowledge, so far none of the solutions to the integration problem has achieved the following goals simultaneously.

1) To exhibit a promising performance with an architecture requiring low memory bandwidth and less implementation price, for example, input-queued switch with FIFO buffers and a speed up of one.
2) To achieve a 100% throughput under different traffic compositions with various percentages of multicast traffic.
3) To utilize the research advances gained for unicast and multicast scheduling, respectively. High-performance queueing policies and scheduling algorithms have been proposed in the context of pure unicast traffic and pure multicast traffic, respectively. Those research results should be utilized.

In a word, the integration of unicast and multicast scheduling is still an open issue. All the above goals are fulfilled by the integration methods presented in Chapter V, which offer an improved performance using an easy to implement architecture.

CHAPTER III

PERFORMANCE ANALYSIS OF LARGE MULTICAST SWITCHES WITH MULTIPLE QUEUES PER INPUT PORT UNDER NON-GATHERED TRAFFIC

HOL blocking compromises the performance of input-queued switches with single FIFO queue per input port. VOQ is used to completely avoid HOL blocking for unicast traffic. Since a multicast packet typically has multiple destined output ports, the exhaustive multicast virtual output queueing is impractical for implementation and results in out-of-order packet delivery. One interesting approach to mitigate HOL blocking for multicast traffic is to allocate a certain number of FIFO queues at each input port. The main concern is how the performance is improved as the number of queues increases.

This chapter theoretically analyzes the performance of large multicast packet switches with multiple FIFO queues per input port under non-gathered Poisson-distributed uniform traffic. The non-gathered traffic comes from all input ports and engages to all output ports. Closed-form expressions are deduced for saturation throughput, average service time, and average delay. Extensive simulations are preformed to verify the theoretical analysis and infer further conclusions. It is shown that a small number of multicast queues (less than ten) are sufficient to maximize saturation throughput and delay performances.

This chapter is organized as follows. Section 3.1 describes the multicast switch architecture used in this chapter. Section 3.2 introduces the initial model and a modified equivalent model. The saturation throughput analysis is given in Section 3.3. In Section 3.4, the average delay and average service time are derived. The theoretical and experimental results are jointly presented in Section 3.5. Finally, Section 3.6 briefly summarizes the content of this chapter.

3.1 Switch Architecture under Non-Gathered Traffic: an $N \times N$ Switch

The architecture of the multicast switch studied in this chapter is shown in Figure 1.

There are N inputs and N outputs with the same link speed. The switch size, N, is assumed to be a large number. At each input port, there are a certain number K of FIFO queues dedicated to multicast traffic. A multicast packet arriving at the input interface is first queued into one of the K multicast queues and then switched from the input port to its target output ports. To preserve the order of packet delivery on the output links, packets of the same flow get queued at the same multicast queue. The switching fabric is an $N \times N$ multicast switch fabric with a speedup of one. In essence, the service discipline is based on fanout splitting [44]. The random scheduling policy [4] resolves contention at the outputs. When more than one HOL packet contend for the same output, one of them is selected randomly with equal probability.

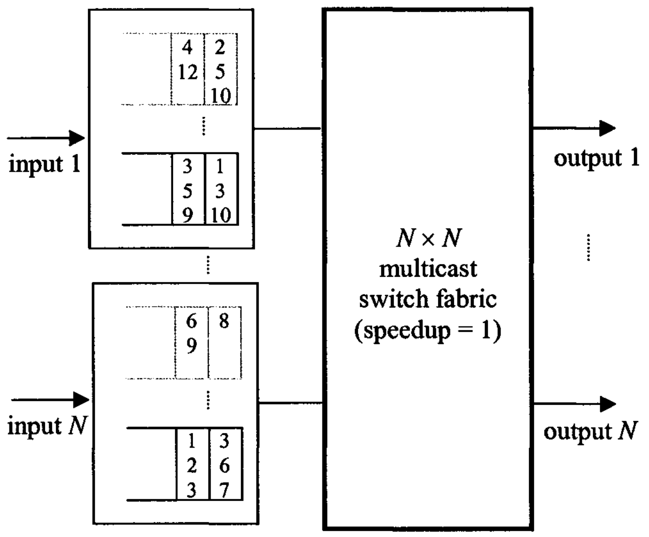


Figure 1. An $N \times N$ multicast switch with K multicast queues per input port.

3.2 Modeling for an $N \times N$ Switch

Some assumptions need to be made before further discussions. The incoming multicast traffic at each input link consists of a mix of multicast flows that follow Poisson distribution. The traffic is uniformly distributed among all inputs. The fanout size of a packet, f, is a random variable with a probability of r_f. The maximal value of f is denoted as F. It is assumed that $F << N$. The f destinations are assumed to be uniformly distributed among the N output ports. As discussed in Chapter I, this assumption is reasonable and does not lose generalization.

3.2.1 Initial Model

The input multicast queues are organized into K groups: all the m-th queues across all input ports belong to group Q^m ($1 \le m \le K$). At an input port, a multicast flow is randomly associated with one of the K queues with equal probability. Packets of a multicast flow get queued to its associated queue. This queueing scheme assures in-order delivery of packets. Since the destinations of a packet are assumed to be uniformly distributed among the N output ports, the Poisson characterization and the uniform distribution properties still apply to each queue.

More than one HOL packets are likely to be available at a single input port. Due to the assumption of a speedup of one, only one of them can be selected for transmission in one time slot. Input contention occurs in this case. It is solved by matching input and output ports in K subsequent rounds within each time slot. In the k-th ($1 \le k \le K$) round, one of the groups that are still unserved is randomly chosen with equal probability. The HOL packets in queues of the selected group are considered for service during this round. The HOL packets of the selected group at the *available* inputs send requests to all its residual destinations. Once an *available* output receives the requests, it randomly grants one of them. As a result, the input receiving the grant and the output initiating the grant get matched. An input (output) port is considered available if it was not matched during the previous rounds within the same time slot. A HOL packet is removed from the input queue when its copies are transmitted to all its target output ports. The number of time slots spent by a packet at the head of its queue is called the *service time* of the HOL packet.

In the initial model above, assuming that an observer samples the state of the $N \times K$ queues at the beginning of each time slot, he sees $N \times K$ M/G/1 queues with identical statistical properties. Because all the input queues served in the k-th round are subject to statistically identical arrival and service processes, the number of packets counted by the observer in any one of the queues has identical statistical properties as in all other queues. This number is defined as the queue length of the k-th round, L_k. Similarly, all the queues in Q^m are also statistically identical. The number of packets counted by the observer in any one of the Q^m queues is defined as the queue length of the m-th group, Γ_m. In the initial model, there is no fixed relationship between the queues served in the k-th round and the queues in group Q^m. Because of the adoption of the random queueing and random serving policies, all groups have identical statistical properties. We have

$$E[\Gamma_i] = E[\Gamma_j] = \Gamma,\ \forall i, j \in [1, K], \tag{1}$$

$$\Pr\{\text{group } Q^m \text{ is served in the } k\text{-th round}\} = \frac{1}{K}. \tag{2}$$

Consequently, we have

$$E\left[L_k\right] = \sum_{m=1}^{K}\left(\Pr\{\text{Group } Q^m \text{ is served in the } k\text{-th round}\} \times E\left[\Gamma_m\right]\right) = \Gamma. \tag{3}$$

Thus, the average queue lengths during any two rounds are always identical.

3.2.2 Modified Model

We now define a model that is logically equivalent to the initial one but easier to be analyzed. In this modified model, we set a fixed relationship between the queues that are served in the k-th round and the queues in group Q^k: the queues in Q^k are always served in the k-th round. As a result, the frequency at which queues in group Q^i are served is obviously higher than the frequency of service of queues in group Q^j if $i < j$. In order for the queue statistics to remain identical over all groups, the arrival rates of multicast packets to queues of different groups must be adjusted accordingly. However, the explicit derivation of the rates of the new Poisson arrival processes is not required, because such rates are irrelevant to the completion of the analysis. The main element of relevance remains the statistical identity and independence of the $N \times K$ M/G/1 queues that compose the model.

The analysis on the saturation throughput and delay performances is based on the modified model. The analytical results will be validated through simulations of the initial model.

3.3 Saturation Throughput Analysis for an $N \times N$ Switch

The following notations hold for the k-th round:

λ_k:	Packet arrival rate for a queue in Q^k;
N_j^k:	Number of HOL packets from the available inputs in Q^k whose residue contains output j;
N'^k_j:	Value of N_j^k in the next time slot;
q_k:	Probability that one destination of the HOL packet in Q^k is serviced in a time slot;
q_k^c:	Conditional probability that one destination of the HOL packet in Q^k is serviced in a time slot given that this input port is available for the k-th round;
X_k:	Service time of the HOL packet in Q^k;
T_k:	Delay of a packet that transits in a queue of Q^k;
A_j^k:	Number of HOL packets arriving in Q^k at the current time slot from the available inputs whose residue contains output j. For a large N, the distribution of A_j^k converges to a Poisson distribution.

3.3.1 Scheduling in the First Round

Each destination of the HOL packets in Q^1 is served independently with an identical probability of q_1, across the inputs as well as from slot to slot. The throughput in the first round (μ_1) is defined as the average number of packets delivered to an output per time slot in the first round. Therefore, $\mu_1 = E[\epsilon(N_j^1)]$, where the indicator function $\epsilon(x) = 1$ if $x > 0$ and $\epsilon(x) = 0$ if $x = 0$.

In the first round, the system behavior is exactly the same as the behavior of a multicast switch with single input queue. Therefore, according to [44], the following three equations hold:

$$q_1 = \frac{2(1-\mu_1)}{2-\mu_1}, \tag{4}$$

$$E[X_1] = \sum_{f=1}^{F} (r_f \times (\sum_{k=1}^{f} \binom{f}{k} \frac{(-1)^{k+1}}{1-(1-q_1)^k})), \tag{5}$$

$$\sum_{f=1}^{F} (f \times r_f) = \mu_{1-sat} \times E[X_1], \tag{6}$$

where μ_{1-sat} is the saturation throughput in the first round. Invoking (4), (5), and (6),

$\mu_{1\text{-}sat}$ can be expressed by substituting μ_1 with $\mu_{1\text{-}sat}$ in (4) and (5).

3.3.2 Scheduling in the Second Round

The destinations of the new arriving HOL packets at the current time slot from the available inputs are randomly distributed over all the output ports. The distributions are uniform and independent of one another. For a large N, the distribution of A_j^2 converges to a Poisson distribution. The throughput in the second round (μ_2) coincides with the average number of packets delivered to an output per time slot in the second round. Notice that output j is matched in the second round if and only if it is not matched in the first round and there is at least one HOL packet in available queues of Q^2 that are destined for it. Therefore, $\mu_2 = E[[1-\epsilon(N_j^1)]\times\epsilon(N_j^2)]$. The dynamic equation for output j is

$$N_j'^2 = N_j^2 - [1-\epsilon(N_j^1)]\times\epsilon(N_j^2) + A_j^2. \tag{7}$$

The first objective is to find the expression for q_2. To this end, the expression for q_2^c need be deduced. In the second round of the scheduling during a particular time slot, there are $\sum_j N_j^2$ destinations at HOL positions of the available input ports, out of which $\sum_j((1-\epsilon(N_j^1))\times\epsilon(N_j^2))$ will be served. Hence, under the assumption of large N,

$$q_2^c = \frac{E[[1-\epsilon(N_j^1)]\times\epsilon(N_j^2)]}{E[N_j^2]}. \tag{8}$$

For the steady state system, through normalization of both sides of (7):

$$E[A_j^2] = E[[1-\epsilon(N_j^1)]\times\epsilon(N_j^2)] = \mu_2. \tag{9}$$

Recalling the definition of $\epsilon(x)$, we have $(\epsilon(x))^2 = \epsilon(x)$ and $x\times\epsilon(x) = x$. Then,

$$([1-\epsilon(N_j^1)]\times\epsilon(N_j^2))^2 = [1-\epsilon(N_j^1)]\times\epsilon(N_j^2), \tag{10}$$

$$N_j^2\times[1-\epsilon(N_j^1)]\times\epsilon(N_j^2) = [1-\epsilon(N_j^1)]\times N_j^2. \tag{11}$$

Because of the assumption of Poisson distribution of A_j^2, we also obtain

$$E[(A_j^2)^2] = (E[A_j^2])^2 + E[A_j^2]. \tag{12}$$

By squaring both sides of (7), substituting (9), (10), (11) and (12), and normalizing both sides, we have

$$E[N_j^2] = \frac{(2-\mu_2)\mu_2}{2(1-\mu_1-\mu_2)}. \tag{13}$$

Substituting (9) and (13) into (8), q_2^c can be expressed as follows:

$$q_2^c = \frac{2(1-\mu_1-\mu_2)}{(2-\mu_2)}. \tag{14}$$

Also,

$$\begin{aligned}
&P\{\text{the input is available for the 2nd round}\}\\
&= \frac{\text{Average number of available inputs for the 2nd round}}{\text{Number of inputs}}\\
&= \frac{E[N - \sum_{j=1}^{N} \epsilon(N_j^1)]}{N}\\
&= \frac{N - N \times E[\epsilon(N_j^1)]}{N}\\
&= 1-\mu_1.
\end{aligned} \tag{15}$$

Based on the relationship between un-conditional probability and conditional probability, $q_2 = P\{\text{the input is available for the second round}\} \times q_2^c$. Together with (14) and (15), we have

$$q_2 = \frac{2(1-\mu_1)(1-\mu_1-\mu_2)}{(2-\mu_2)}. \tag{16}$$

According to the analysis result in [14], the average service time of the HOL packet served in the second round is given as below:

$$E[X_2] = \sum_{f=1}^{F} (r_f \times (\sum_{k=1}^{f} \binom{f}{k} \frac{(-1)^{k+1}}{1-(1-q_2)^k})). \tag{17}$$

According to the **P-K** formula [14, 15], the average delay for a packet being transiting in a queue of Q^2 can be expressed as

$$E[T_2] = E[X_2] + \frac{\lambda_2 \times E[X_2^2]}{2(1-\lambda_2 \times E[X_2])}. \tag{18}$$

From (18), we can see that the average delay becomes infinite when $1 = \lambda_2 E[X_2]$.

Considering that $\mu_2 = \lambda_2 \times \sum_{f=1}^{F}(f \times r_f)$ and (17), for the saturation throughput in the second round, μ_{2-sat}, we have:

$$\sum_{f=1}^{F}(f \times r_f) = \mu_{2-sat} \times \sum_{f=1}^{F}(r_f \times (\sum_{k=1}^{f}\binom{f}{k}\frac{(-1)^{k+1}}{1-(1-q_2)^k})). \qquad (19)$$

Using (19) together with (16), where μ_1 and μ_2 are substituted with μ_{1-sat} and μ_{2-sat}, respectively, μ_{2-sat} can be calculated. As an example, the theoretical results of the saturation throughput (μ_{sat}) for a multicast switch with two queues and a constant fanout size of F ($r_f = 0$, $\forall f \in [1, F-1]$; and $r_F = 1$) are listed in Table 1, where $\mu_{sat} = \mu_{1-sat} + \mu_{2-sat}$.

Table 1. Saturation throughput for switches with two queues per input port.

F	1	2	4	8	10
λ_1	0.586	0.695	0.779	0.849	0.868
λ_2	0.127	0.092	0.068	0.047	0.041
λ	0.713	0.787	0.847	0.896	0.910

3.3.3 Scheduling in the k-th Round

In this subsection, analysis is focused on the throughput and service time at the k-th round ($2 \le k \le K$). The throughput in the k-th round (μ_k) is defined as the average number of packets delivered to an output per time slot in the k-th round. During each round, the switch could be modeled as N identical and independent $M/G/1$ queues, where each queue is associated with a distinct switch output. The total switch throughput is the sum of the respective throughputs achieved in K rounds: $\mu = \sum_{k=1}^{K}\mu_k$. Notice that output j is matched in the k-th round if and only if it is not matched in the previous rounds and at least one HOL packet in available queues of Q^k is destined for it. Therefore, $\mu_k = E[\prod_{i=1}^{k-1}(1-\epsilon(N_j^i)) \times \epsilon(N_j^k)]$. The dynamic equation for output j is

$$N'^k_j = N^k_j - \prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times \epsilon(N^k_j) + A^k_j. \qquad (20)$$

At first, the expression for q_k need be found. To this end, the expression for q^c_k need be deduced. In the k-th round of the scheduling during a particular time slot, there are $\sum_j N^k_j$ destinations at HOL positions of the available input ports, out of which $\sum_j(\prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times \epsilon(N^k_j))$ will be served. Hence, for a large N, we have

$$q^c_k = \frac{E[\prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times \epsilon(N^k_j)]}{E[N^k_j]}. \qquad (21)$$

Taking the average of both sides of (20) and considering that for the steady–state system $E[N'^k_j] = E[N^k_j]$, we have

$$E[A^k_j] = E[\prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times \epsilon(N^k_j)] = \mu_k. \qquad (22)$$

Similar to (12), we have

$$E[(A^k_j)^2] = (E[A^k_j])^2 + E[A^k_j]. \qquad (23)$$

Taking into account the definition of ϵ, we have

$$N^k_j \times \prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times \epsilon(N^k_j) = \prod_{i=1}^{k-1}(1-\epsilon(N^i_j)) \times N^k_j. \qquad (24)$$

Since $k \ll N$, the N^i_j values $(1 \le i \le k)$ are independent of each other. We can then derive the following equation:

$$E[\prod_{i=1}^{k-1}(1-\epsilon(N^i_j))] = 1 - \sum_{i=1}^{k-1}\mu_i. \qquad (25)$$

Proof for (25): When $k = 2$, (25) is $E[1-\epsilon(N^1_j)] = 1-\mu_1$, which is correct according to the analysis given in Section 3.3.1. Assuming that (25) is correct when $k = l$, i.e., $E[\prod_{i=1}^{l-1}(1-\epsilon(N^i_j))] = 1 - \sum_{i=1}^{l-1}\mu_i$.

Since $\mu_l = E[\prod_{i=1}^{l-1}(1-\epsilon(N^i_j)) \times \epsilon(N^l_j)] = E[\prod_{i=1}^{l-1}(1-\epsilon(N^i_j))] \times E[\epsilon(N^l_j)]$, then

$$E[1-\epsilon(N_j^l)] = 1 - \frac{\mu_l}{E[\prod_{i=1}^{l-1}(1-\epsilon(N_j^i))]} = 1 - \frac{\mu_l}{1-\sum_{i=1}^{l-1}\mu_i} = \frac{1-\sum_{i=1}^{l}\mu_i}{1-\sum_{i=1}^{l-1}\mu_i}.$$

Therefore, $E[\prod_{i=1}^{l+1-1}(1-\epsilon(N_j^i))] = E[\prod_{i=1}^{l-1}(1-\epsilon(N_j^i))] \times E[1-\epsilon(N_j^l)] = 1-\sum_{i=1}^{l}\mu_i$, i.e., (25) is also correct when $k = l + 1$.

In sum, (25) was proved for $2 \le k \le K$ and the proof is complete. □

In addition, $(\prod_{i=1}^{k-1}(1-\epsilon(N_j^i)) \times \epsilon(N_j^k))^2 = \prod_{i=1}^{k-1}(1-\epsilon(N_j^i)) \times \epsilon(N_j^k)$. Combining this result with (22), (23), (24), and (25), and squaring and normalizing both sides of (20), we obtain:

$$E[N_j^k] = \frac{(2-\mu_k)\mu_k}{2(1-\sum_{i=1}^{k}\mu_i)}. \tag{26}$$

Substituting (22) and (26) into (21), we reach the following result:

$$q_k^c = \frac{2 \times (1-\sum_{i=1}^{k}\mu_i)}{(2-\mu_k)}. \tag{27}$$

Given the assumption of randomly granting to one of the requests at each output,

P\{the input is available for the k-th round\}

$$= \frac{\text{Average number of available inputs for the } k\text{-th round}}{\text{Number of inputs}}$$

$$= \frac{E[N - \sum_{i=1}^{k-1}(\sum_{j=1}^{N}\prod_{l=1}^{i-1}(1-\epsilon(N_j^l)) \times \epsilon(N_j^i))]}{N}$$

$$= \frac{N - N \times \sum_{i=1}^{k-1} E[\prod_{l=1}^{i-1}(1-\epsilon(N_j^l)) \times \epsilon(N_j^i)])}{N}$$

$$= 1 - \sum_{i=1}^{k-1}\mu_i. \tag{28}$$

Also, q_k = P\{the input is available for the k-th round\} × q_k^c. Combining this result together with (27) and (28), we obtain

$$q_k = \frac{2\times(1-\sum_{i=1}^{k-1}\mu_i)\times(1-\sum_{i=1}^{k}\mu_i)}{(2-\mu_k)}. \tag{29}$$

Similar to (17), the average service time in the k-th round is given by

$$E[X_k] = \sum_{f=1}^{F}(r_f \times (\sum_{i=1}^{f}\binom{f}{i}\frac{(-1)^{i+1}}{1-(1-q_k)^i})). \tag{30}$$

Based on the **P-K** formula [14, 15], the average delay for a packet transiting in a queue of Q^k can be expressed as

$$E[T_k] = E[X_k] + \frac{\lambda_k \times E[X_k^2]}{2\times(1-\lambda_k \times E[X_k])}. \tag{31}$$

We can see that the average delay for a queue in the k-th round becomes infinite when $1=\lambda_k E[X_k]$, where $\mu_k = \lambda_k \times \sum_{f=1}^{F}(f\times r_f)$. Hence, recalling (30), the saturation throughput in the k-th round, μ_{k-sat}, satisfies the following equation:

$$\sum_{f=1}^{F}(f\times r_f) = \mu_{k-sat} \times \sum_{f=1}^{F}(r_f \times (\sum_{i=1}^{f}\binom{f}{i}\frac{(-1)^{i+1}}{1-(1-q_k)^i})). \tag{32}$$

Together with (29), where μ_i is substituted with μ_{i-sat} ($1 \le i \le k$) and μ_{i-sat} ($1 \le i \le k-1$) are calculated during previous rounds, μ_{k-sat} can be calculated.

As an example, the theoretical results of the saturation throughput (μ_{sat}) for a multicast switch with three queues and a constant fanout size of F are listed in Table 2, where $\mu_{sat} = \mu_{1-sat} + \mu_{2-sat} + \mu_{3-sat}$.

Table 2. Saturation throughput for switches with three queues per input port.

F	1	2	4	8	10
μ_{sat}	0.586	0.695	0.779	0.849	0.868
λ_2	0.127	0.092	0.068	0.047	0.041
λ_3	0.066	0.048	0.035	0.024	0.021
λ	0.779	0.835	0.882	0.921	0.931

3.4 Delay Analysis for an $N \times N$ Switch

The following notations are used in the analysis of the delay performance:

μ: The output load at an output link;

T: The delay of a packet in any queue;

N_k: The queue length of the queue in Q^k.

When the overall packet arrival rate is less than the saturation packet arrival rate, i.e., before the system delay becomes infinite, the queueing system is stable and the following condition holds:

$$\mu = \sum_{k=1}^{K} \mu_k . \tag{33}$$

According to [14], we have

$$E[X_k^2] = \sum_{f=1}^{F} (r_f \times (\sum_{i=1}^{f} \binom{f}{i} (-1)^{i+1} \frac{2(1-q_k)^i}{(1-(1-q_k)^i)^2})) + E[X_k], \tag{34}$$

where q_k and $E[X_k]$ are given by (29) and (30), respectively.

According to Little's theorem [15], we obtain

$$E[L_k] = \lambda_k \times E[T_k]. \tag{35}$$

As stated earlier, due to the equivalence between the initial and modified models, the average queue lengths of any two queues in the modified model are all identical:

$$E[L_1] = E[L_2] = \dots = E[L_K]. \tag{36}$$

We also have $\mu_k = \lambda_k \times \sum_{f=1}^{F} (f \times r_f)$. Combining this relationship with (29), (30), (31), and (33) through (36), μ_k and $E[T_k]$ can be calculated for a given output load, μ. The resulting average system delay is as follows:

$$E[T] = \frac{\sum_{k=1}^{K} \mu_k \times E[T_k]}{\sum_{k=1}^{K} \mu_k}. \tag{37}$$

As an example, let us consider a multicast switch with three queues per input buffer ($K = 3$). Assuming that all multicast packets have a constant fanout size of four, we substitute different values of μ ($0 < \mu < 1$) into the derived equations. $E[T_k]$ ($k = 1, 2, 3$)

and the average system delay $E[T]$ are calculated and plotted in Figure 2. As we expected, $E[T_1] < E[T_2] < E[T_3]$ follows for all admissible output load. The validity of the delay analysis will be further verified by simulations.

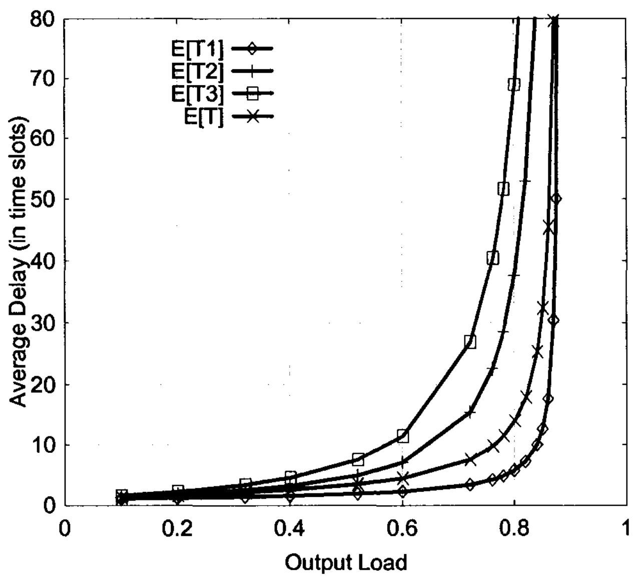


Figure 2. Average delay as a function of normalized output load with $K = 3$ and a constant fanout size of four.

3.5 Numerical Results of Theoretical Analysis and Simulations

Extensive simulations were performed for different switch sizes and fanout sizes to verify the analysis results and to infer further conclusions. Based on both theoretical and simulated results, the performance characteristics of large multicast packet switches with multiple FIFO queues per input port are discussed in this subsection. The duration of all simulation runs is one million time slots. Data are collected for statistical elaboration during the last half million time slots. Infinite queue size is assumed to avoid packet loss. The fanout size is a constant of F, i.e., $r_f = 0$, $\forall f \in [1, F-1]$; and $r_F = 1$. The simulation results are obtained on a system that reflects the initial model of a multicast switch, which is implemented in a switch simulator SIM++ [54].

3.5.1 Verification to Analytical Results via Simulations

Figure 3 shows the theoretical and simulation saturation throughput as a function of fanout size for multicast switches with K queues ($K = 2$, 3, and 4) per input port. The simulation saturation throughput is determined as follows. The throughput is kept increasing until achieving a maximal match size, which remain constant regardless how big the output load is. The maximal match size divided by N is then the saturation throughput for the given N and F. The discrepancy between analysis and simulation results is always far below 2%, which confirms the accuracy of the analysis.

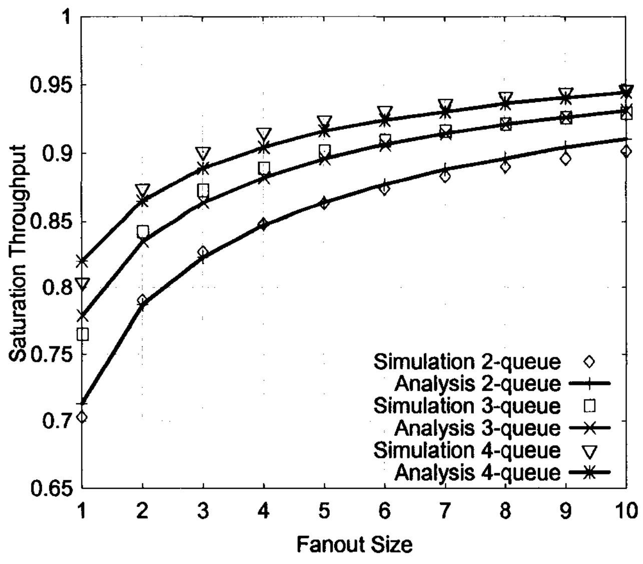


Figure 3. Saturation throughput as a function of fanout size.
(The switch size used in these simulations is 1024 x 1024.)

Figures 4 and 5 show the simulation results for the saturation throughput as a function of switch size in the presence of two and three queues per input port. The dash lines show the corresponding theoretical saturation throughputs. The purpose of these two plots is to highlight the convergence of the saturation throughput to its asymptotic value as the switch size increases, for different fanout sizes. One can see that the convergence is faster for smaller fanout sizes. The saturation throughput remains fairly constant for $N > 80$ in all scenarios. For $N > 80$ and bigger fanout sizes, the difference between simulation results and respective theoretical results under the assumption of a very big N becomes indistinguishable, hinting that the analytical results are indeed valid for practical switches that lie in the upper end ($N > 80$) of the size scale. It should be noticed that we can only claim that the difference between the simulation saturation throughputs and theoretical saturation throughputs is very close to each other. It is hard to justify which one is bigger than the other.

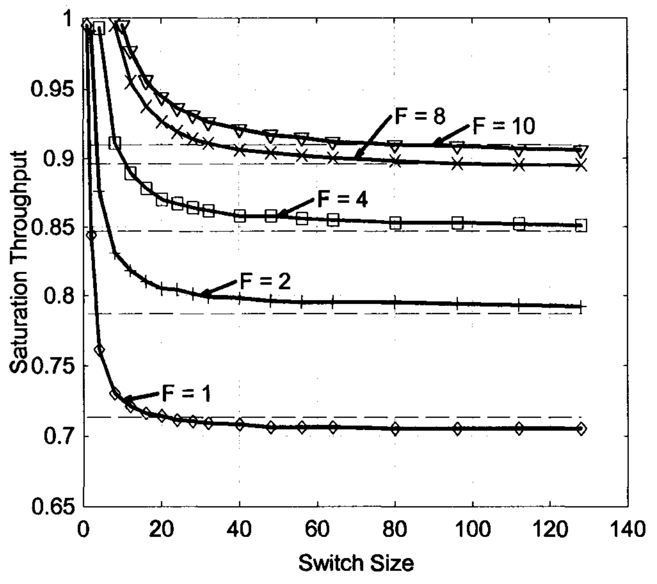


Figure 4. Saturation throughput as a function of switch size with $K = 2$.

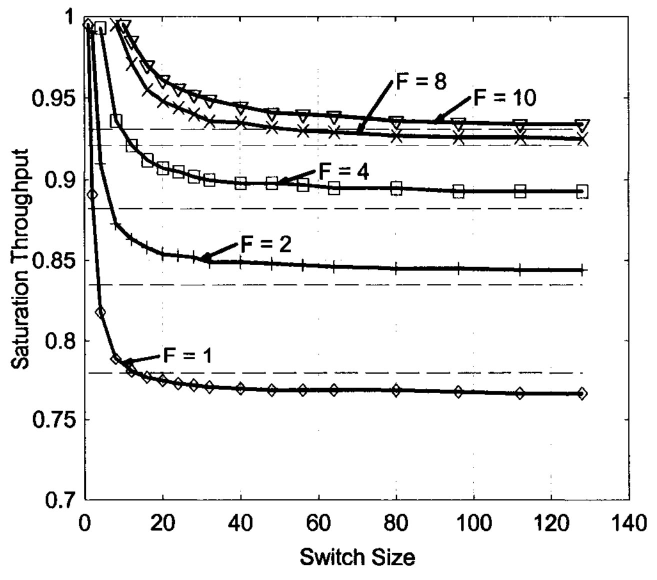


Figure 5. Saturation throughput as a function of switch size with $K = 3$.

Simulated and analytical average delays are plotted in Figures 6, 7, and 8, as a function of the output load offered to a 256 × 256 switch with K = 2, 3, and 4, respectively. Each plot corresponds to a different value of F (F = 2, 4, 8). Simulation results and theoretical results always agree well.

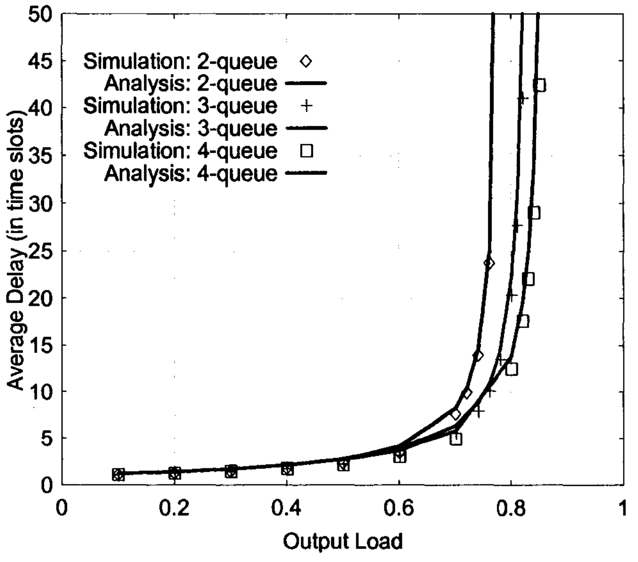


Figure 6. Average delay as a function of normalized output load with $F = 2$. (The switch size used in these simulations is 256 × 256.)

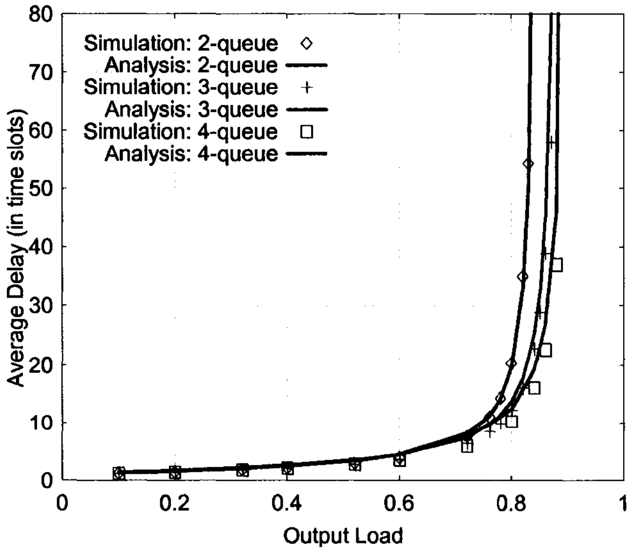


Figure 7. Average delay as a function of normalized output load with $F = 4$. (The switch size used in these simulations is 256 × 256.)

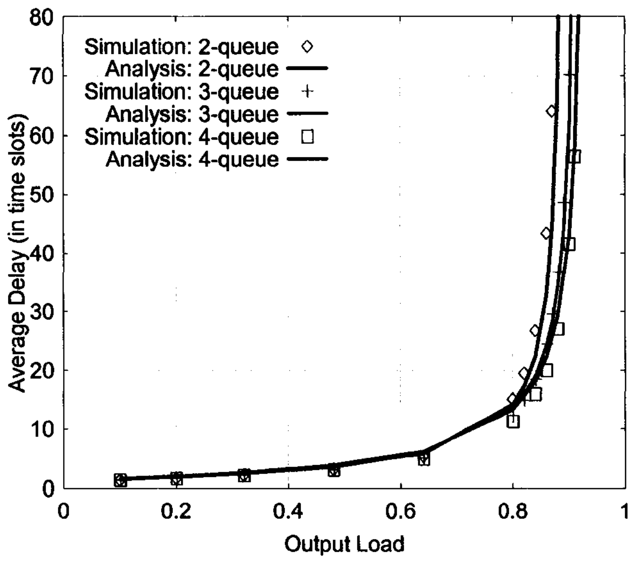


Figure 8. Average delay as a function of normalized output load with $F = 8$.

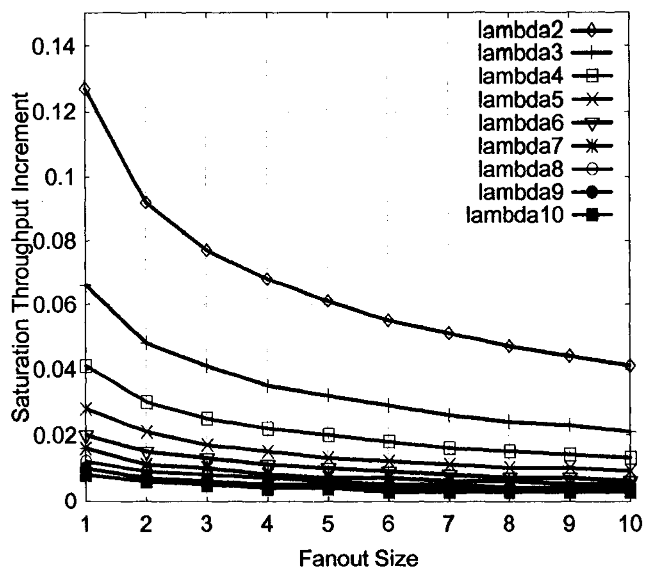
(The switch size used in these simulations is 256 × 256.)

The match between the simulation data and the theoretical data shown in Figures 3 through 8 testifies to the correctness of the analysis under the assumptions of the modified model. In addition, based on the results shown in Figures 6, 7, and 8, a number of interesting conclusions can be drawn. When the output load is much less than the saturation throughput, which indicates the switch capacity, there is no obvious difference among the average delay of multicast switches with two queues, three queues and four queues per input port. However, as the output load increases and approaches the saturation throughput, the average delay difference between multicast switches with different numbers of queues per input port becomes more and more obvious. Furthermore, using the equations derived in the previous part of this chapter, it is possible to calculate the number of queues per input port needed in order to meet specific delay requirements for a multicast switch where no prior knowledge of the input and output distribution of multicast traffic is available. Under the same conditions, when the number of queues per input port exceeds a certain threshold the gain in delay performance guaranteed by additional queues is not obvious.

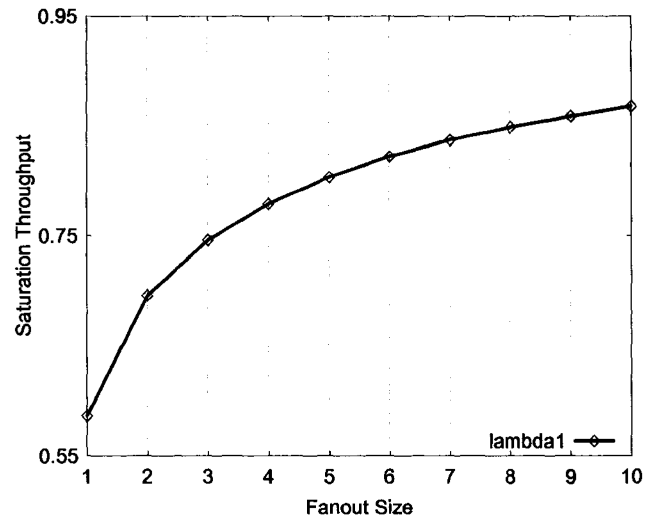
3.5.2 Performance Improvement Gained by Adding More Queues per Input Port

Figure 9(a) shows the increment of saturation throughput μ_k ($k = 2, ..., 10$) by adding the k-th queue at every input as a function of F. Figure 9(b) illustrates the saturation throughput μ_1 of a multicast switch with one queue per input port. One can see that the bigger the value of F, the bigger the value of μ_1. This is because a big value of F alleviates the HOL blocking and thus increases the throughput. Meanwhile an increasing value of μ_1 results in a decreasing space in the subsequent iterations. Thus, the bigger the value of μ_1, the smaller the value of μ_k ($k = 2, ..., 10$).

It is also observed that while the saturation throughput of a multicast switch increases with the number of queues per input port, the benefit of additional queues becomes less and less sensible as new queues get added. In the particular case of Figure 9, the throughput contribution of the tenth queue approaches zero when the overall saturation throughput with nine queues is already above the 90% mark. For clarity, the analytical values of $\mu = \sum_{k=1}^{9} \mu_k$ and μ_{10} under different fanout sizes are listed in Table 3.



(a)



(b)

Figure 9. Increment of saturation throughput by adding the k-th ($k = 2, \ldots, 10$) queue as a function of fanout sizes.

Table 3. Theoretical saturation throughput of a switch with $K = 9$ and the increment by adding the tenth queue.

f	2	4	6	8	9	10
λ	0.930	0.950	0.961	0.967	0.969	0.972
λ_{10}	0.006	0.004	0.003	0.003	0.003	0.003

Therefore, it can be concluded that high throughput can be achieved in a large multicast switch with a few queues (less than 10) per input port if multicast traffic is uniformly distributed over all inputs and outputs. To corroborate this conclusion, Figure 10 plots the theoretical values of saturation throughput as the number of queues per input port ranges from 1 to 10. The benefit of using multiple queues per input port is initially obvious, but fades as the number of queues per input port grows larger.

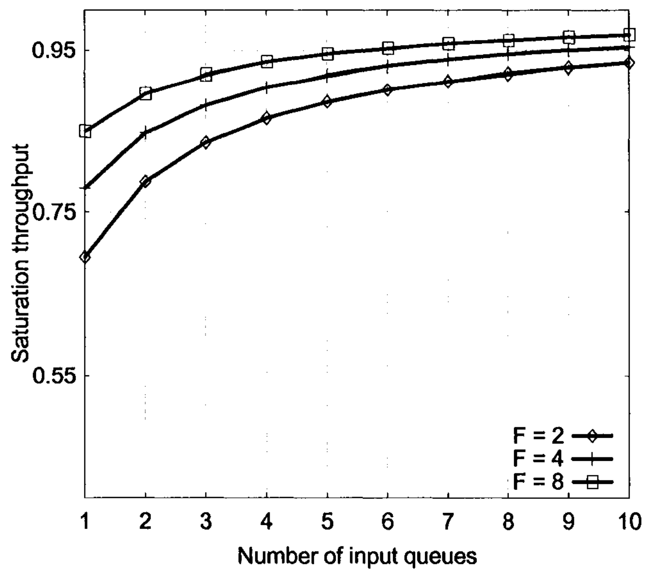


Figure 10. Saturation throughput as a function of the number of queues per input port.

The last set of simulation experiments are run on a 256 × 256 multicast switch with K equals to 1, 2, 3, 9, and 10, respectively. Figures 11, 12, and 13 show the delay as a function of the output load with $F = 2$, 4, and 8, respectively. Since the log-scale for the y-axis is adopted in order to show the result clearly, these curves are slightly s-shaped. The delay difference is obvious between the cases with one, two, and three queues. Conversely, the difference of delay performance becomes indistinguishable between switches with nine and ten multicast queues per input port buffer.

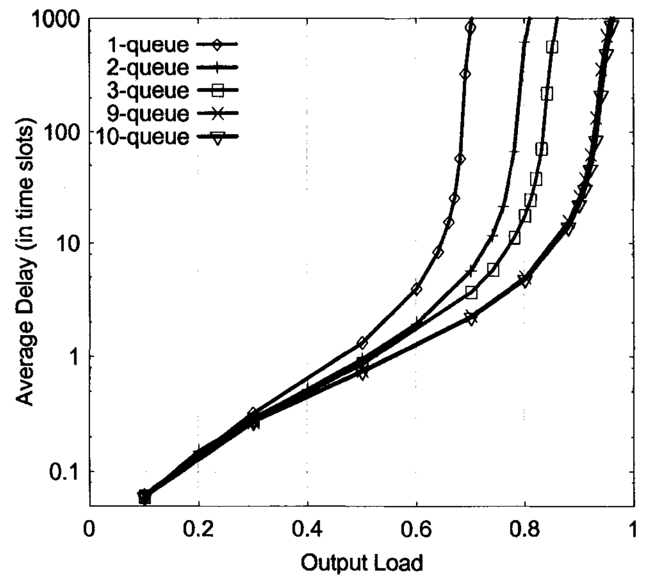


Figure 11. Delay as a function of normalized output load with $F = 2$.

(The switch size used in these simulations is 256 × 256.)

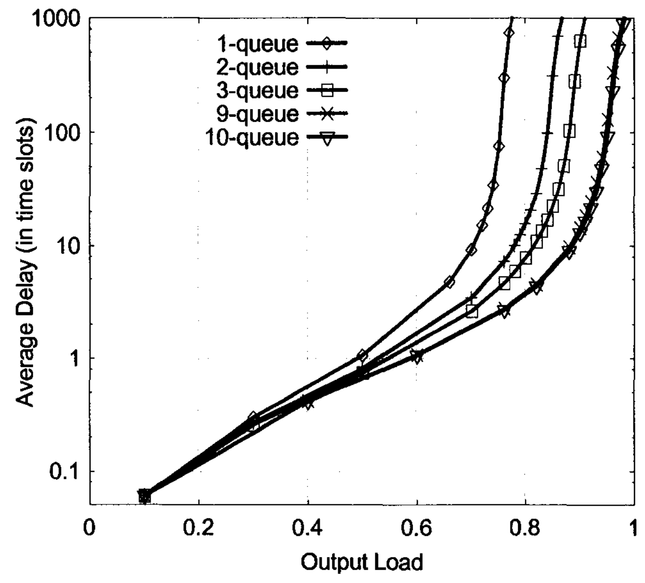


Figure 12. Delay as a function of normalized output load with $F = 4$.

(The switch size used in these simulations is 256 × 256.)

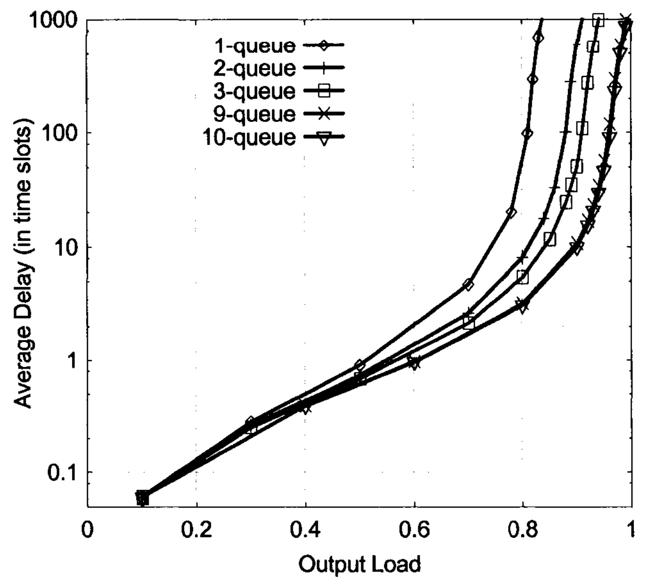


Figure 13. Delay as a function of normalized output load with $F = 8$.

(The switch size used in these simulations is 256 × 256.)

3.6 Summary

Using the $M/G/1$ model and the queueing theory, this chapter analyzed the performance of multicast switches with multiple queues per input port under the assumption of uniform traffic distribution over all inputs and outputs. The closed-form expressions for the saturation throughput, the average service time, and the average delay were provided. Extensive simulations were performed. The results show that the throughput and delay performance of the multicast switch improves as the number of queues per input buffer increases. In a large multicast switch, the performance indices converge to their asymptotic values when the number of input queues is relatively small. Thus, a small number of input queues per input buffer (no more than 10) are sufficient to achieve sub-optimal performance in large switches with uniformly distributed multicast traffic.

CHAPTER IV

PERFORMANCE ANALYSIS OF LARGE MULTICAST SWITCHES WITH MULTIPLE QUEUES PER INPUT PORT UNDER GATHERED TRAFFIC

In Chapter III, the saturation throughput and delay performances of a large $N \times N$ multicast switch with multiple FIFO queues per input port are theoretically analyzed. This chapter generalizes the theoretical analysis to a large $M \times N$ multicast switch, i.e., the number of input ports may be different from the number of output ports. This generalization is motivated by the importance of the *gathered-traffic* scenario. Multicast applications often generate sustained and long-lasting flows, which may gather among fewer input ports and engage more output ports at a given router or switch [4]. Therefore, the gathered-traffic scenario is widely adopted in the simulation studies on multicast queueing and scheduling [4, 12, 13, 44].

This chapter presents the theoretical analysis on the saturation throughput and delay performances of a large $M \times N$ multicast switch with multiple FIFO queues per input port, and extensive simulation studies for validation. A Markov chain model is proposed to deduce the probability distribution function of residue size at the beginning of a time slot and analyze the availability of an input after a certain number of iterations' competition for service. It is shown that firstly a small number of queues, which is much less than $2^N - 1$, is a reasonable choice for the tradeoff between the saturation throughput and delay performances and the scheduling overhead; secondly, the final achievable saturation throughput decreases as the ratio of M/N decreases; and thirdly, the analysis results are indeed valid for practical large-sized switches.

This chapter is organized as follows. Section 4.1 describes switch architecture and modeling. The availability of an input in the k-th round is analyzed based on a model of Markov chain in Section 4.2. Saturation throughput analysis and delay analysis are given in Sections 4.3 and 4.4, respectively. Section 4.5 presents both theoretical and experimental results. Finally, this chapter is briefly summarized in Section 4.6.

4.1 Generalization to an $M \times N$ Switch: Switch Architecture and Modeling

The architecture of an $M \times N$ multicast packet switches considered in this chapter is shown in Figure 14. There are M inputs and N outputs with the same link speed. Both M and N are assumed to be large numbers with M/N being a certain value. At each input port, there are K FIFO queues dedicated to multicast traffic. The switching fabric is an $M \times N$ multicast switch fabric with a speedup of one. The service discipline is based on fanout splitting.

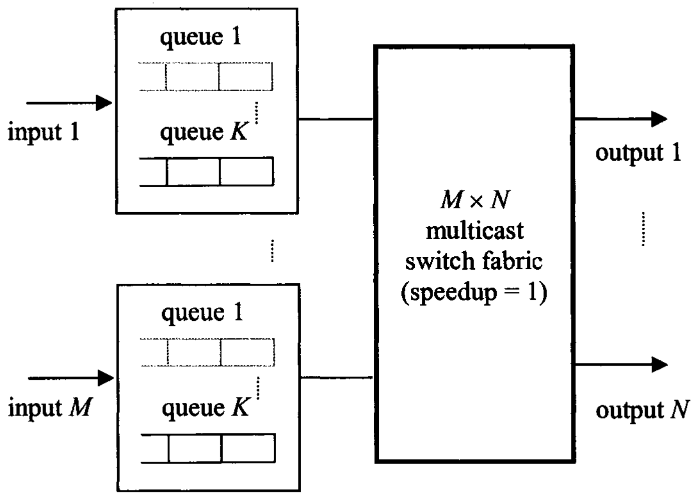


Figure 14. An $M \times N$ switch with K multicast FIFO queues per input port.

While the switch architecture is generalized from an $N \times N$ switch to an $M \times N$ switch, other assumptions, the random queueing policy, and the random scheduling algorithm that are adopted in the performance analysis on the $N \times N$ switch still holds. To make it clear, they are briefly restated as following:

- The incoming multicast traffic follows Poisson distribution.
- The traffic is uniformly distributed among all inputs.
- The fanout size of a packet, f, is a random variable with a probability of r_f. The maximal value of f is denoted as F. $F << N$ and the f destinations are uniformly distributed among the N output ports.
- Random queueing policy: At an input port, a multicast flow is randomly

associated with one of the K queues with equal probability. Packets of a multicast flow get queued to its associated queue.

- Random scheduling algorithm: All the m-th queues across all input ports form the group Q^m ($1 \leq m \leq K$). Input and output ports are matched in K subsequent rounds within each time slot. In the k-th ($1 \leq k \leq K$) round, one of the groups that are still unserved is randomly chosen to be served with equal probability. The HOL packets of the selected group at the *available* inputs send requests to all its residual destinations. In each round, when more than one HOL packets compete for an *available* output port, one of them are granted randomly. An input (output) port is considered available if it was not matched during the previous rounds within the same time slot. The number of time slots spent by a packet at the head of its queue is called the *service time* of the HOL packet.

In the modified model, a fixed relationship is set up between the queues that are served in the k-th round and the queues in group Q^k: the queues in Q^k are always served in the k-th round. Similarly to the result about the average queue length in the k-th round, L_k, given in Chapter III, the average queue lengths during any two rounds are always identical, i.e.,

$$E[L_1] = E[L_2] = \ldots = E[L_K]. \tag{38}$$

4.2 Availability of an Input Port in the *k*-th Round

The following notations hold for the k-th round.

λ_k:	The packet arrival rate for a queue in Q^k.
q_k:	The probability that one destination of the HOL packet in Q^k is serviced in each time slot.
X_k:	The service time of the HOL packet in Q^k.
u_k^0:	The probability that a queue in Q^k is empty when a HOL packet leaves that queue.

In this subsection, the main objective is to discuss the probability that an input is not matched in the previous rounds of the k-th round, i.e., the probability that an input is available for the k-th round.

Considering that a queue belonging to Q^k is an $M/G/1$ queue, based on the analysis to a model named "Imbedded Markov Chain" in [41], we have

$$u_k^0 = 1 - \lambda_k E[X_k]. \tag{39}$$

Let $R_n^{(k)} = R^{(k)}(t_n)$ represent the residue size of the HOL packet at the beginning of the n-th time slot in a given input queue during the k-th round. The matching decision in each round is made according to the output packets competing in current time slot. The past of the residue size has no influence on the future if the present is specified, i.e.,

$$P\{R_{n+1}^{(k)} \mid R_n^{(k)}, R_{n-1}^{(k)}, \ldots, R_0^{(k)}\} = P\{R_{n+1}^{(k)} \mid R_n^{(k)}\}. \tag{40}$$

Therefore, $R_n^{(k)}$ is a discrete Markov chain with finite states. The set of states is {0, 1, 2, ..., F}, where state 0 represents the case that the queue is empty and state i ($1 \le i \le F$) represents the case that the residue size of the HOL packet equals to i. This Markov chain is homogenous and its one step transition matrix, P_k, is given by

$$P_k = \begin{bmatrix} p_{0,0}^{(k)} & p_{0,1}^{(k)} & \cdots & p_{0,F}^{(k)} \\ p_{1,0}^{(k)} & p_{1,1}^{(k)} & \cdots & p_{1,F}^{(k)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{F,0}^{(k)} & p_{F,1}^{(k)} & \cdots & p_{F,F}^{(k)} \end{bmatrix}, \tag{41}$$

where $p_{i,j}^{(k)} \triangleq P\{R_{n+1}^{(k)} = j \mid R_n^{(k)} = i\}$.

For a given HOL packet with $R_n^{(k)}$ equaling to i, r destinations may be served in the n-th time slot. If $r < i$, the remaining $(i - r)$ destinations continue competing for service in the $(n+1)$-th time slot and $R_{n+1}^{(k)} = r - l$. If $r = i$, all residue destinations of the HOL packet are served and this packet is removed from the input queue at the end of n-th time slot. At the beginning of $(n+1)$-th time slot, if the queue is empty, $R_{n+1}^{(k)} = 0$; otherwise a new HOL packet is waiting for service and $R_{n+1}^{(k)}$ equals to its fanout, f. The arrival of a packet in a time slot is independent of the number of packets stored in the queue. Also, the number of destinations of a HOL packet that are served in a time slot is also independent of the number of packets stored in a queue when a HOL packet leaves. Therefore, the values of $p_{i,j}^{(k)}$ are calculated as following.

1. $i = 0$:

   $p_{0,0}^{(k)} = P$\{no packet arrives in a time slot | queue is empty\}

$$= P\{\text{no packet arrives in a time slot}\} = 1-\lambda_k;$$

$$p_{0,j}^{(k)} = P\{\text{a packet arrives in a time slot} \mid \text{queue is empty}\} \times P\{f=j\}$$

$$= P\{\text{a packet arrives in a time slot}\} \times r_j$$

$$= \lambda_k r_j, \qquad \forall j \in [1,F];$$

2. $1 \le i \le F$:

$$p_{i,0}^{(k)} = P\{i \text{ destinations of a HOL packet are served}\}$$

$$\times P\{\text{queue is empty when a HOL packet leaves}\}$$

$$= \binom{i}{0} \times (1-q_k)^0 \times {q_k}^i \times u_k^0 = u_k^0 {q_k}^i;$$

$$p_{i,j}^{(k)} = P\{(i-j) \text{ destinations of a HOL packet are served}\}$$

$$+ P\{i \text{ destinations of a HOL packet are served and queue is not empty when a HOL packet leaves}\} \times P\{f=j\}$$

$$= \binom{i}{j}(1-q_k)^j {q_k}^{i-j} + (1-u_k^0){q_k}^i r_j, \qquad \forall j \in [1,i];$$

$$p_{i,j}^{(k)} = P\{i \text{ destinations of a HOL packet are served and queue is not empty when a HOL packet leaves}\} \times P\{f=j\}$$

$$= (1-u_k^0){q_k}^i r_j, \qquad \forall j \in [i+1,F].$$

Then the one-step transition function is as follows:

$$P_k = \begin{pmatrix} 1-\lambda_k & \lambda_k r_1 & \cdots & \lambda_k r_F \\ u_k^0 q_k & (1-q_k)+(1-u_k^0)q_k r_1 & \cdots & (1-u_k^0)q_k r_F \\ u_k^0 {q_k}^2 & \binom{2}{1}(1-q_k)q_k + (1-u_k^0){q_k}^2 r_1 & \cdots & (1-u_k^0){q_k}^2 r_F \\ \vdots & \vdots & \vdots & \vdots \\ u_k^0 {q_k}^i & \binom{i}{1}(1-q_k){q_k}^{i-1} + (1-u_k^0){q_k}^i r_1 & \cdots & (1-u_k^0){q_k}^i r_F \\ \vdots & \vdots & \vdots & \vdots \\ u_k^0 {q_k}^{F-1} & \binom{F-1}{1}(1-q_k){q_k}^{F-2} + (1-u_k^0){q_k}^{F-1} r_1 & \cdots & (1-u_k^0){q_k}^{F-1} r_F \\ u_k^0 {q_k}^F & \binom{F}{1}(1-q_k){q_k}^{F-1} + (1-u_k^0){q_k}^F r_1 & \cdots & (1-q_k)^F + (1-u_k^0){q_k}^F r_F \end{pmatrix}, \quad (42)$$

where u_k^0 is given by (39).

According to Perron's theorem [41], we have

$$\lim_{n\to\infty} P\{\boldsymbol{R}_n^{(k)} = i\} = v_k^i, \qquad \forall i \in [0, F]. \tag{43}$$

$v_k^0, v_k^1, \ldots,$ and v_k^F are called as limiting probabilities and they can be calculated by solving the following equations:

$$v_k = v_k \times P_k, \tag{44}$$

$$\sum_{j=0}^{F} v_k^j = 1, \tag{45}$$

where P_k is given by (42), and $v_k = [v_k^0 \quad v_k^1 \quad \cdots \quad v_k^F]$.

If and only if none of the destinations of an HOL packet at a given input is served in the k-th round, it is said that this input is not matched during the k-th round. The probability that an input is not matched during the k-th round is denoted as σ_k. We have

$$\begin{aligned}\sigma_k &= \sum_{i=0}^{F} (P\{\text{None of } i \text{ destinations is served in } k\text{-th round}\} \times \lim_{n\to\infty} P\{\boldsymbol{R}_n^{(k)} = i\}) \\ &= \sum_{i=0}^{F} \binom{i}{0} (1-q_k)^i q_k{}^0 v_k^i \\ &= \sum_{i=0}^{F} (1-q_k)^i v_k^i .\end{aligned} \tag{46}$$

If and only if an input is not matched during all the previous rounds, this input is available for the k-th round. Therefore, we have

$$\begin{aligned} &P\{\text{input is available for the } k\text{th round}\} \\ &= 1 - \sum_{i=1}^{k-1} P\{\text{an input is matched in the } i\text{th round}\} \\ &= 1 - \sum_{i=1}^{k-1} (1-\sigma_i) \\ &= \sum_{i=1}^{k-1} \sigma_i - k + 2 .\end{aligned} \tag{47}$$

4.3 Saturation Throughput Analysis for an $M \times N$ Switch

The following notations are used in further discussions of this chapter.

q_k^c: The conditional probability that one destination of the HOL packet in Q^k is serviced in each time slot given that this input port is available for the k-th round.

T_k: The delay of the packet transited in a queue of Q^k.

A_j^k: The number of HOL packets arriving in Q^k at the current time slot from the available inputs whose set of destinations contains output j. For a large N, the distribution of A_j^k converges to a Poisson distribution.

μ_k: The throughput in the k-th round.

4.3.1 Scheduling in the First Round

In a stable $M \times N$ Switch, the following relationship is satisfied:

$$\mu_1 = \frac{M}{N} \times \lambda_1 \times \sum_{f=1}^{F} (f \times r_f). \tag{48}$$

In terms of the throughput in the first round, the system behavior is exactly the same as the behavior of a multicast switch with single input queue that was discussed in [14]. Therefore, according to [14], the following three equations hold:

$$q_1 = \frac{2(1-\mu_1)}{2-\mu_1}, \tag{49}$$

$$E[X_1] = \sum_{f=1}^{F} (r_f \times (\sum_{k=1}^{f} \binom{f}{k} \frac{(-1)^{k+1}}{1-(1-q_1)^k})), \tag{50}$$

$$\frac{M}{N} \times \sum_{f=1}^{F} (f \times r_f) = \mu_{1\text{-}sat} \times E[X_1], \tag{51}$$

where $\mu_{1\text{-}sat}$ is the saturation throughput in the first round. Combining (49) through (51) together, where μ_1 is substituted with $\mu_{1\text{-}sat}$, $\mu_{1\text{-}sat}$ can be calculated.

4.3.2 Scheduling in the k-th Round

Similarly to (48), in a stable $M \times N$ Switch, we have

$$\mu_k = \frac{M}{N} \times \lambda_k \times \sum_{f=1}^{F} (f \times r_f). \tag{52}$$

Given that this input port is available for the k-th round, the conditional probability (q_k^c) that one destination of the HOL packet in Q^k is served in a time slot is expressed as

$$q_k^c = \frac{2\times(1-\sum_{i=1}^{k}\mu_i)}{(2-\mu_k)}, \tag{53}$$

the deduction of which is the same as that of (27) given in Chapter III and hence is omitted here. Combining (46) with (47), we have

$$\text{Pr \{the input is available for the } k\text{-th round\}} = \sum_{h=1}^{k-1}(\sum_{i=0}^{F}(1-q_h)^i v_h^i) - k + 2. \tag{54}$$

Since $q_k = P$\{the input is available for the k-th round\} $\times$ q_k^c, together with (53) and (54), we obtain

$$q_k = (\sum_{h=1}^{k-1}(\sum_{i=0}^{F}(1-q_h)^i v_h^i) - k + 2)\frac{2(1-\sum_{i=1}^{k}\mu_i)}{(2-\mu_k)}, \tag{55}$$

where v_h^0, v_h^1, ... and v_h^F are expressed by solving (44) and (45) with $k = h$.

According to the analysis results in [14], the average service time in the k-th round is given by

$$E[X_k] = \sum_{f=1}^{F}(r_f \times (\sum_{i=1}^{f}\binom{f}{i}\frac{(-1)^{i+1}}{1-(1-q_k)^i})). \tag{56}$$

Based on the **P-K** formula [14, 15], the average delay for a packet transiting in the k-th round can be expressed as

$$E[T_k] = E[X_k] + \frac{\lambda_k \times E[X_k^2]}{2(1-\lambda_k \times E[X_k])}. \tag{57}$$

We can see that the average delay for a queue in the k-th round becomes infinite if $1 = \lambda_k E[X_k]$. Then, recalling (52) and (56), for the saturation throughput in the k-th round, μ_{k-sat}, we have

$$\frac{M}{N}\times\sum_{f=1}^{F}(f\times r_f) = \mu_{k-sat}\times\sum_{f=1}^{F}(r_f \times (\sum_{i=1}^{f}\binom{f}{i}\frac{(-1)^{i+1}}{1-(1-q_k)^i})). \tag{58}$$

Together with (55), where μ_i is substituted with μ_{i-sat} ($1 \le i \le k$) and μ_{i-sat} ($1 \le i \le k-1$) are calculated during previous rounds, μ_{k-sat} can be calculated.

As an example, the numeric theoretical results of the saturation throughput (μ_{sat}) for a multicast switch with four queues, a given value *M*/*N*, and a constant fanout size of *F* are listed in Table 4, where $\mu_{sat} = \mu_{1-sat} + \mu_{2-sat} + \mu_{3-sat} + \mu_{4-sat}$.

Table 4. Saturation throughput for switches with four queues per input port.

M/*N*	5/8				¾			
F	2	4	6	8	2	4	6	8
$\mu_{1\text{-sat}}$	0.58	0.68	0.74	0.78	0.62	0.72	0.77	0.81
$\mu_{2\text{-sat}}$	0.08	0.09	0.09	0.09	0.10	0.10	0.10	0.09
$\mu_{3\text{-sat}}$	0.04	0.04	0.04	0.04	0.05	0.05	0.04	0.04
$\mu_{4\text{-sat}}$	0.02	0.02	0.02	0.02	0.03	0.03	0.02	0.02
μ_{sat}	0.72	0.83	0.89	0.93	0.80	0.90	0.93	0.96

4.4 Delay Analysis for an *M* × *N* Switch

The following notations are used in the analysis on the delay performance.

μ:	The output load at an output link.
T:	The delay of a packet in any queue.
L_k:	The queue length of the queue in Q^k.

When the overall packet arrival rate is less than the saturation packet arrival rate, the queueing system is stable and the following relationship holds:

$$\mu = \sum_{k=1}^{K} \mu_k . \tag{59}$$

According to [14], we have

$$E[X_k^2] = \sum_{f=1}^{F} (r_f \times (\sum_{i=1}^{f} \binom{f}{i} (-1)^{i+1} \frac{2(1-q_k)^i}{(1-(1-q_k)^i)^2})) + E[X_k], \tag{60}$$

where q_k and $E[X_k]$ are given by (55) and (56), respectively. Based on Little's theorem [15], we obtain

$$E[L_k] = \lambda_k \times E[T_k]. \tag{61}$$

Combining (38), (52), (55) through (57), and (59) through (61), μ_k and $E[T_k]$ can be calculated for a given output load. The average system delay is expressed as follows:

$$E[T] = \frac{\sum_{k=1}^{K} \mu_k \times E[T_k]}{\sum_{k=1}^{K} \mu_k}. \tag{62}$$

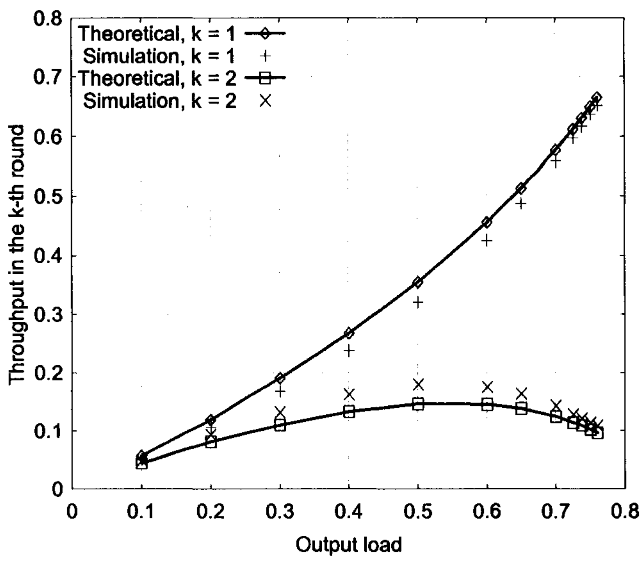


Figure 15. Throughput in the *k*-th round as a function of normalized output load with $M/N = 5/8$, $F = 4$, $K = 2$.

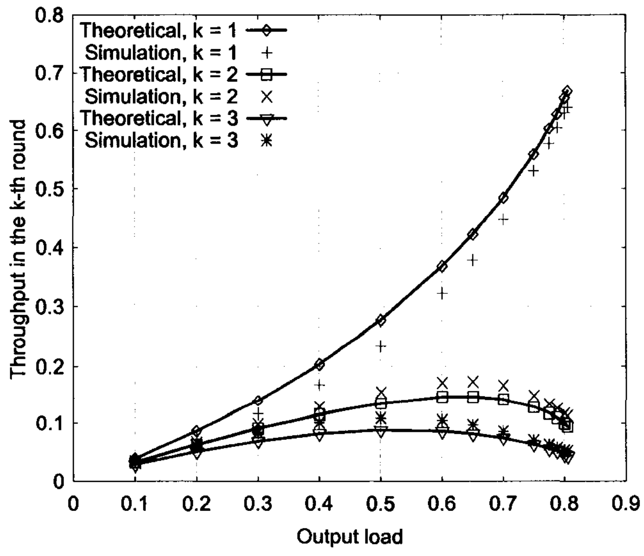


Figure 16. Throughput in the *k*-th round as a function of normalized output load with $M/N = 5/8$, $F = 4$, $K = 3$.

From (62), we can see that the delay analysis relies on the prediction on the throughput in the k-th round, μ_k ($k = 1, 2, \ldots, K$), under a given output load. The relationship given in (38) determines the distribution of the overall output load among K rounds. To double check this formula, simulations are performed for a 640×1024 multicast switch with a constant fanout size of F. Figures 15 and 16 plot the theoretical and simulated throughput in the k-th round, μ_k, as a function of the output load for $K = 2$ and 3, respectively. We can see that the simulated results agree with the theoretical results well.

4.5 Numerical Results of Theoretical Analysis and Simulations

Extensive simulations were performed for different switch sizes, values of M/N, and fanout sizes to verify the analysis results and to infer further conclusions. Based on both theoretical and simulated results, the performance characteristics of large multicast packet switches with multiple FIFO queues per input are discussed. The duration of all simulation runs is one million time slots. Data are collected for statistical elaboration during the last half million time slots. Infinite queue size is assumed to avoid packet loss. The fanout size is a constant of F ($r_f = 0$, $\forall f \in [1, F-1]$; and $r_F = 1$). The simulation results are obtained on a system that reflects the initial model of a multicast switch, which is implemented in a switch simulator SIM++ [54].

4.5.1 Simulation Verification

In order to verify the theoretical analysis, simulations are performed for switches with $N = 1024$. Figure 17 shows the theoretical and simulated saturation throughput as a function of K under different values of M/N and F. The discrepancy between analysis and simulation results is always below 2% under all scenarios, which confirms the accuracy of the analysis. Figures 18 and 19 plot the theoretical and simulated average delay as a function of output load for switches with K FIFO queues per input port ($K = 2$, 3, 4, and 5) for the case that $M/N = 3/4$ and $F = 3$ and the case that $M/N = 5/8$ and $F = 4$, respectively. We can see that the simulated average delays always agree well with the theoretical results. The match between the simulation data and the theoretical data testifies to the equivalence between the modified model and the initial model, and the correctness of the theoretical analysis. In addition, based on both theoretical and simulated numerical results, some interesting performance characteristics are going to be discussed in the following subsections.

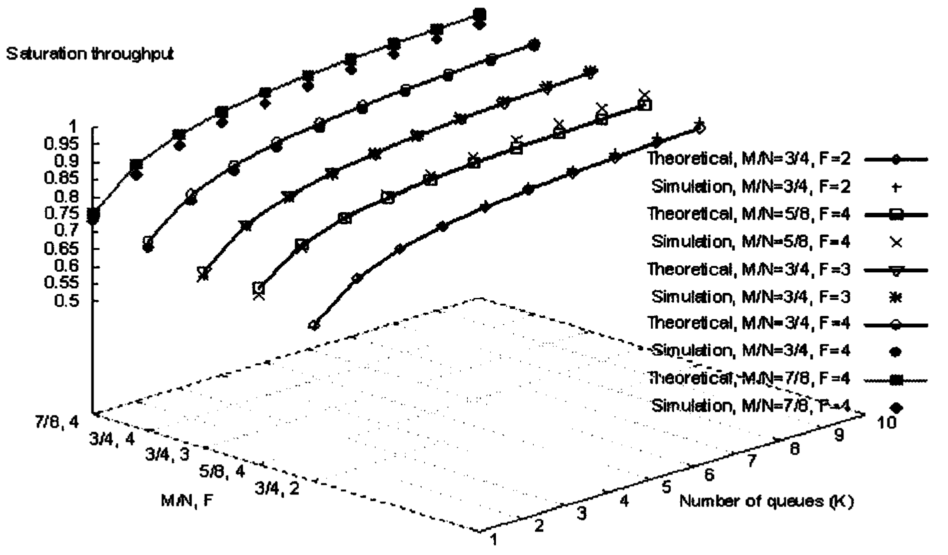


Figure 17. Saturation throughput as a function of K.

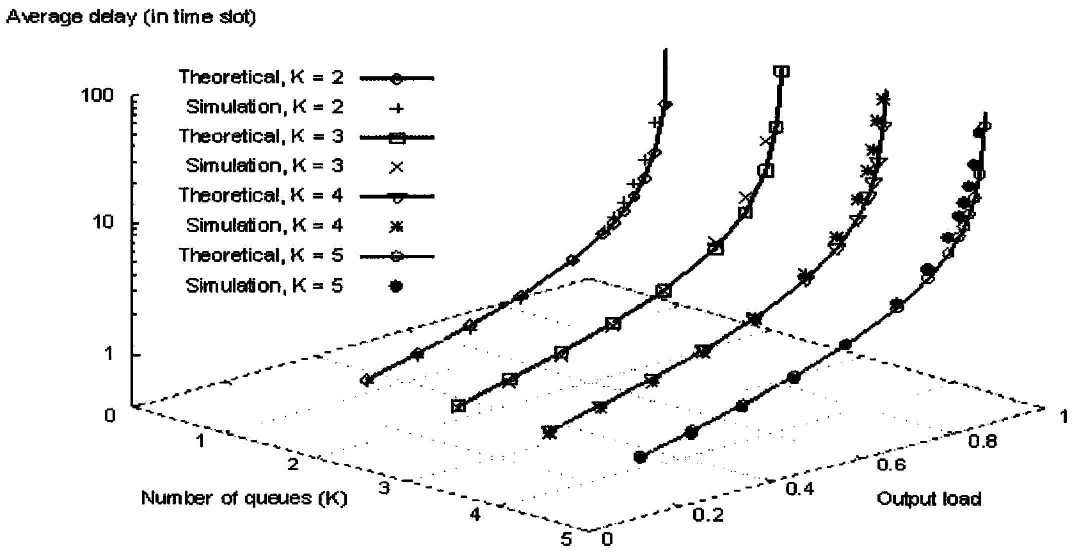


Figure 18. Delay as a function of normalized output load with $M/N = 3/4$ and $F = 3$.

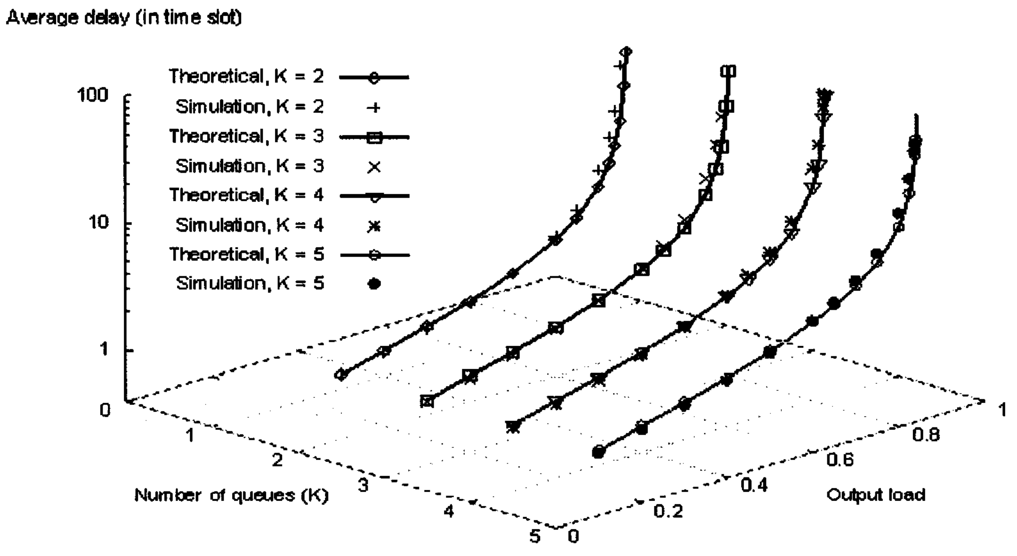


Figure 19. Delay as a function of normalized output load with $M/N = 5/8$, $F = 4$.

4.5.2 Convergence of Saturation Throughput as N Increases

During the theoretical analysis, it is assumed that both N and M are very big numbers. Then the question is how big the value of N should be so that the theoretical analysis is valid. Figures 20 and 21 illustrate the convergence of the saturation throughput as the increment of N under different values of M/N and F for switches with two and four FIFO multicast queues per input port, respectively. The dash lines show the corresponding theoretical saturation throughput. One can see that the saturation throughput converges to its asymptotic value as N increases and it remains fairly constant for $N > 80$ in all scenarios. This result indicates that the analytical results are indeed valid for practical switches that lie in the upper end ($N > 80$) of the size scale. It should be noticed that we can only claim that the theoretical and simulated saturation throughputs are very close to each other. It is hard to justify which one is bigger.

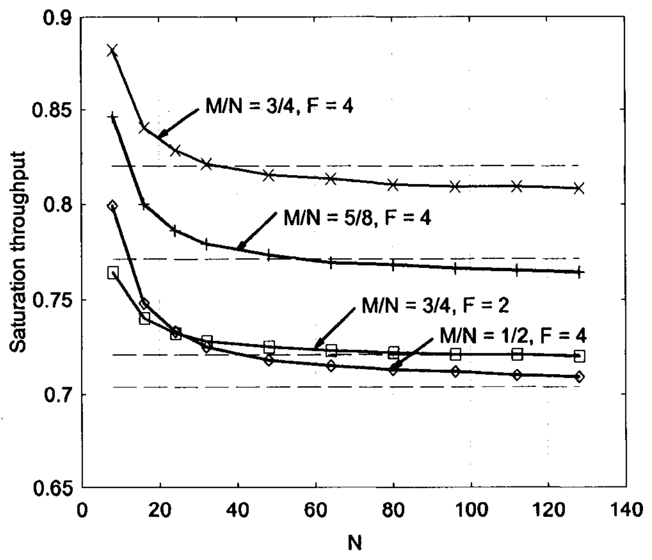


Figure 20. Saturation throughput as a function of N with $K = 2$.

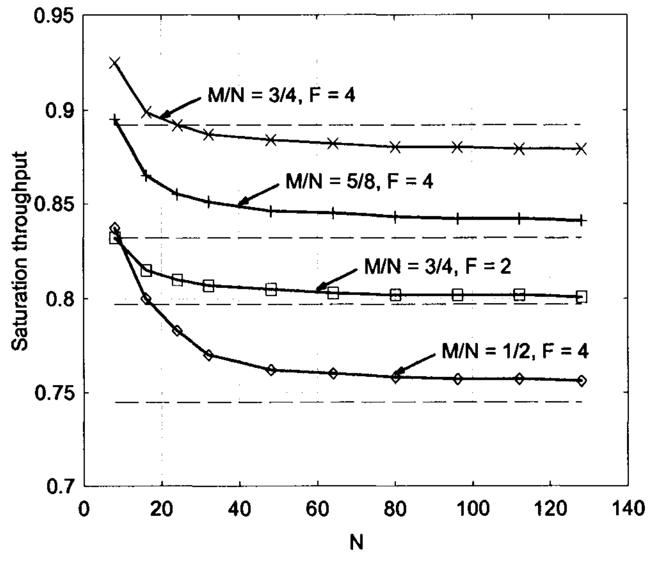


Figure 21. Saturation throughput as a function of N with $K = 4$.

4.5.3 Performance Improvement by Increasing K

In high-speed packet switches, high performance is pursued twofold. From the packet transmission point of view, the packets arriving at their input ports should be delivered to the corresponding output ports as efficiently as possible, i.e., high throughput and small delay are expected. On the other hand, from the scheduling overhead point of view, the time used to make scheduling decisions should be as short as possible, especially for very high-speed switches with very short time slots. By increasing *K*, the throughput and delay performance can be improved. However, as the increment of *K*, the scheduling overhead increases since each queue needs to be scanned. The number of queues per input port should be a trade-off between the throughput and delay performances and the scheduling overhead. Therefore, this subsection is going to investigate how the throughput and delay performance is improved by increasing *K*.

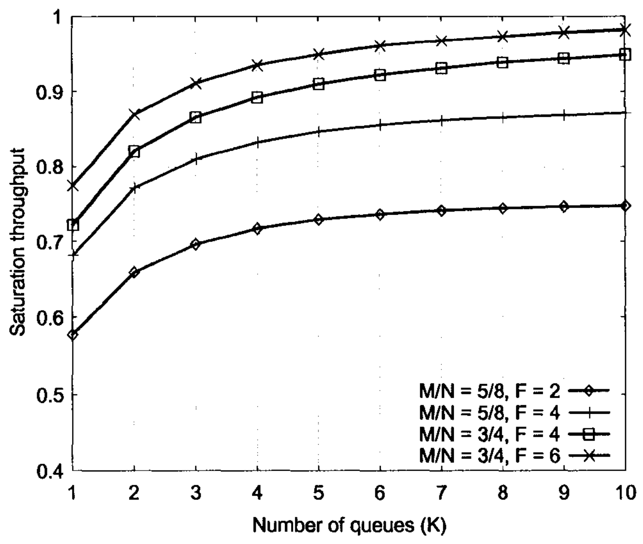


Figure 22. Saturation throughput as a function of *K*.

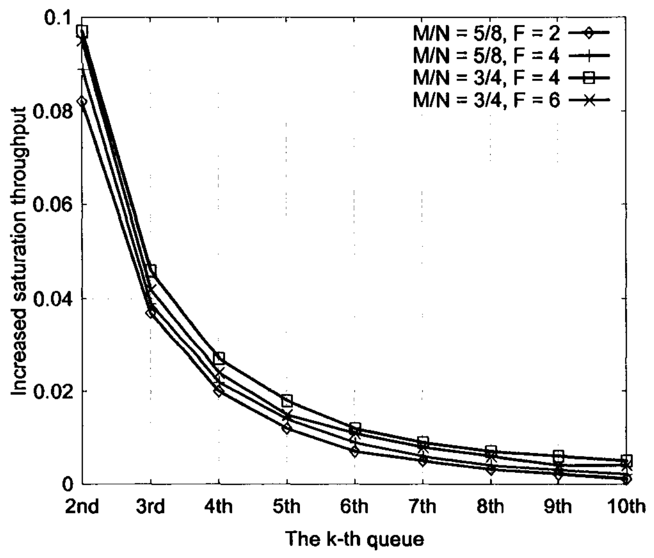


Figure 23. Saturation throughput increment by adding the *k*-th queue.

The theoretical saturation throughput as a function of K ($K = 1, 2, \ldots, 10$) is shown in Figure 22. The increased saturation throughput by adding the k-th ($k = 2, 3, \ldots, 10$) queue is illustrated in Figure 23. One can see that the increment of saturation throughput by adding the k-th queue is significant when k is small. The performance improvement introduced by addition queues becomes less and less sensible as new queues are added although the saturation throughput keeps increasing as K increases. This observation is further confirmed by the simulation study to the delay performance. Figures 24 and 25 plot the average delay as a function of output load for an 80 × 128 switch under a constant fanout size of two and a 96 × 128 switch under a constant fanout size of six, respectively. The curve of average delay shifts to right as the increment of K, which indicates the improvement of the delay performance. Similar to the saturation throughput performance, the delay performance is improved significantly by adding new queues when K is small, and the benefit becomes less and less as K increases. Considering the scheduling overhead, the reasonable number of queues would be a small number, which is much less than $2^N - 1$. Using the equations given in the previous part of this chapter, the necessary number of queues can be calculated based on saturation throughput and/or delay requirements.

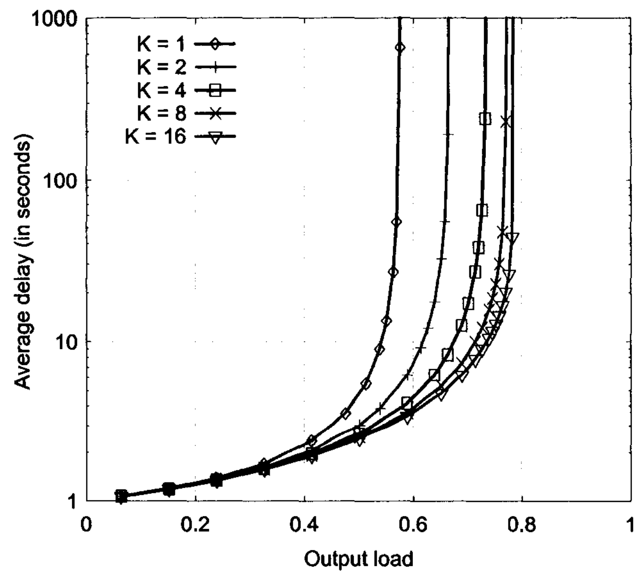


Figure 24. Delay as a function of normalized output load for an 80 × 128 switch with $F = 2$.

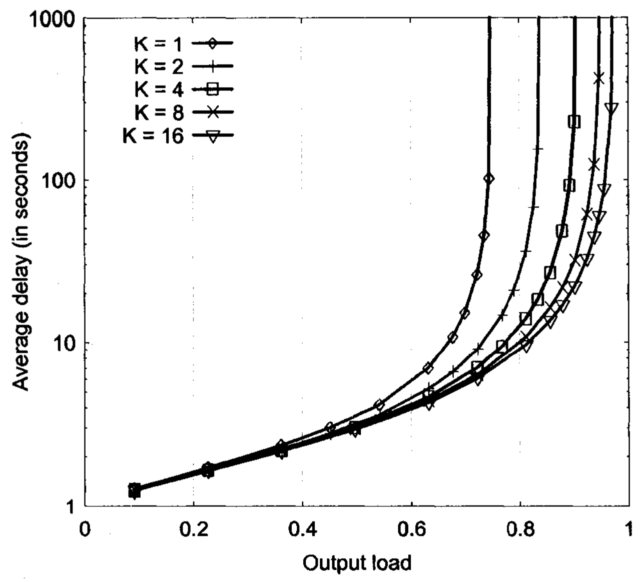


Figure 25. Delay as a function of normalized output load for a 96 × 128 switch with $F = 6$.

4.5.4 Contention at the Input Ports

The performance of a switch is also impacted by how much the traffic gathers among input ports, i.e., the ratio of M to N. The theoretical saturation throughput as a function of K under different values of M/N is illustrated in Figure 26. With a given number of queues, the saturation throughput decreases as the ratio of M to N decreases. The simulated average delay as a function of output load is plotted in Figure 27. The delay performance is also degraded as the ratio of M to N decreases, which is consistent with the degradation of the saturation throughput performance. This result can be explained as follows. With a given N, the number of available inputs in each round is smaller when the traffic gathers at fewer input ports. Considering that one packet at most can be scheduled for transmission at an input port in each time slot, it is not surprising that the performance is degraded as the ratio of M to N decreases.

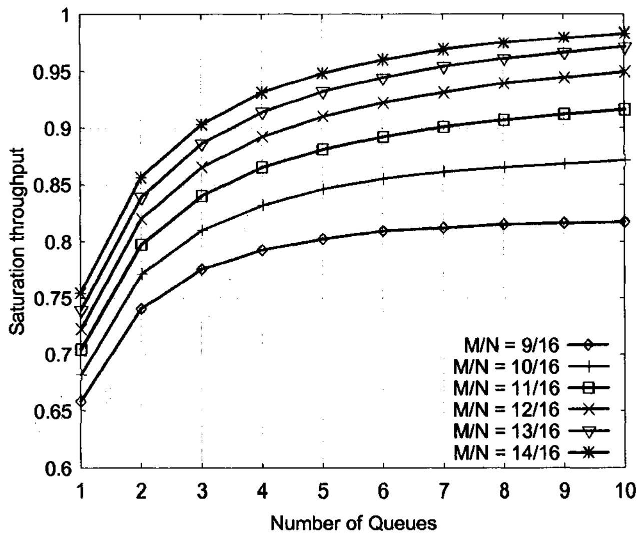


Figure 26. Saturation throughput as a function of K with $F = 4$.

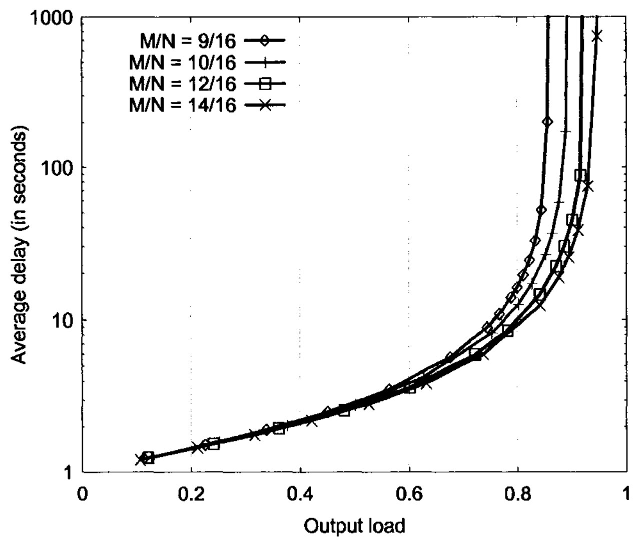


Figure 27. Delay as a function of normalized output load with $N = 128$, $F = 4$, and $K = 8$.

4.6 Summary

In this chapter, using the model of $M/G/1$ and queueing theory, the performance of a large $M \times N$ multicast switch with multiple FIFO queues per input port were theoretically analyzed. A model of Markov chain was proposed to analyze the availability of an input after a certain number of iterations' competition for service. The closed-form expressions for saturation throughput, average service time, and average delay were deduced. Extensive simulations were performed to further verify the theoretical analysis. These closed-form expressions can be used to decide the necessary number of queues per input port according to the design requirements on saturation throughput and/or delay.

Based on both theoretical and simulated numerical results, several observations were concluded. Firstly, in terms of the saturation throughput and the average delay, the performance of a multicast switch can be significantly improved by increasing the number of queues per input port when the number of queues is small. However, the improvement drops quickly as the increment of the number of queues. Therefore, a small number of queues, which is much less than $2^N - 1$, are a reasonable choice for the tradeoff between the saturation throughput and delay performances and the scheduling overhead. Secondly, due to the input contention, the performance of a multicast switch decreases when the traffic gathers among fewer input ports. Thirdly, the analytical results are indeed valid for practical large-sized switches.

CHAPTER V

INTEGRATION OF UNICAST AND MULTICAST SCHEDULING WITHIN ONE SWITCHING FABRIC

Packet queueing and scheduling have been extensively studied in the context of either pure unicast traffic or pure multicast traffic. Unfortunately, the results from a study in one context are not applicable to the other context. The design of integrated scheduling for both types of traffic remains an open issue. This chapter deals with the problem of integrating unicast and multicast scheduling in an $N \times N$ input-queued packet switch with first in first out buffers. Instead of using isolated switching fabrics for unicast and multicast traffic respectively, one switching fabric is efficiently utilized for both unicast and multicast traffic by a careful design. In the design, each input port maintains a set of unicast queues based on virtual output queueing technique and a set of multicast queues based on a load-balance policy [4]. Two practical integration algorithms, loosely slot-coupled integration algorithm (*l*SCIA) and fully slot-coupled integration algorithm (*f*SCIA) are proposed. Both theoretical analysis and simulation studies suggest that the proposed integrated scheduling algorithms exhibit a promising performance in terms of throughput, delay and packet loss ratio, at different traffic compositions.

This chapter is organized as follows. Section 5.1 specifies the switch architecture, the traffic models, and the performance metrics. *l*SCIA and *f*SCIA are proposed and their properties are discussed in Section 5.2. The theoretical analysis on the selection of multicast service ratio is presented in Section 5.3. Section 5.4 provides the simulated performance analysis. Finally, a brief summary is given in Section 5.5.

5.1 Switch Architecture and Traffic Models

5.1.1 The Switch Architecture

The proposed switch model is an input-queued switch shown in Figure 28. Two sets of queues are organized separately at each input port. For unicast traffic, VOQ technique is deployed; for multicast traffic, a small number of FIFO queues are allocated at each input port, which is based on the research results achieved in Chapter III. Unicast packets are assigned to the proper queue according to their destinations. The task of assigning multicast packets to the appropriate queues is not straightforward since the number of multicast queues (K) is much smaller than the number of possible destination vectors. In this paper, possible destination vectors are partitioned into K groups based on the load-balance policy [4] such that output load is equalized across groups. Each group is associated with a multicast queue. An incoming multicast packet is assigned to one of the multicast queues according to its destination vector. Such an assigning policy assures in-order packet deliver. The input buffer size is a finite. Once the input buffer is full, the new incoming packet will be dropped. The switching fabric is an $N \times N$ multicast switch fabric with a speedup of one.

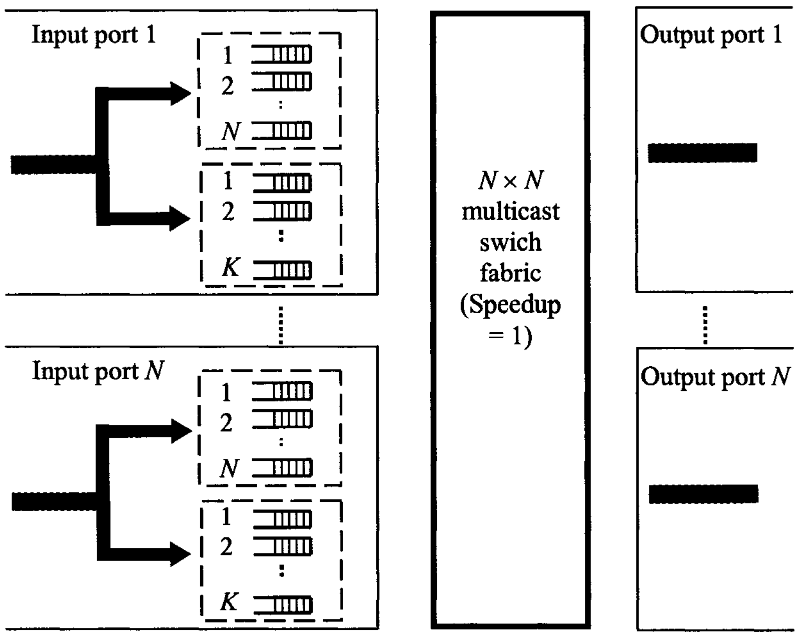


Figure 28. Switch architecture for hybrid traffic.

5.1.2 Traffic Models

The input traffic consists of both unicast and multicast traffic, which are mixed together with a certain arrival rate ratio. Given an input i, the link arrival rate is denoted as λ_i; the arrival rates of unicast and multicast traffic are denoted as $\lambda_{i\text{-}u}$ and $\lambda_{i\text{-}m}$, respectively. We have

$$\lambda_{i-u} = P_u \times \lambda_i, \text{ and } \lambda_{i-m} = (1 - P_u) \times \lambda_i, \tag{63}$$

where P_u represents the probability that an arrival packet is a unicast packet.

If $\lambda_i = \lambda_j$, $\forall\ i, j \in [1, N]$, the traffic is uniform traffic and the average input arrival rate, λ, equals to λ_i. Otherwise, the input traffic is non-uniform traffic and $\lambda = \frac{1}{N}\sum_{i=1}^{N}\lambda_i$.

Let λ_u and λ_m be the average unicast and multicast arrival rates, respectively, then $\lambda_u = \frac{1}{N}\sum_{i=1}^{N}\lambda_{i-u}$ and $\lambda_m = \frac{1}{N}\sum_{i=1}^{N}\lambda_{i-m}$.

The fanout size of a multicast packet, F, is a random variable uniformly distributed within $[F_{min}, F_{max}]$. The F destined output ports of a multicast packet are randomly selected among all output ports with equal probability. The total number of possible fanout sets, N_{FV}, equals to $\sum_{f=F_{\min}}^{F_{\max}}\binom{N}{f}$. The fanout set of a multicast packet is randomly chosen among all possible fanout sets. Let $E[F]$ be the average fanout size, then

$$E[F] = \sum_{f=F_{\min}}^{F_{\max}} \frac{f \times \binom{N}{f}}{N_{FV}}. \tag{64}$$

Output load is proportional to the corresponding traffic arrival rate. Due to the uniform distribution of packet destinations, there is no need to distinguish which output port it is. The output load is denoted as μ, and the unicast and multicast output loads are denoted as μ_u and μ_m, respectively. The following relations hold:

$$\mu_u = \lambda_u, \tag{65}$$

$$\mu_m = \lambda_m \times E[F], \tag{66}$$

$$\mu = \mu_u + \mu_m = \lambda \times P_u + \lambda \times (1 - P_u) \times E[F]. \tag{67}$$

Only admissible traffic is considered, i.e., $0 < \lambda < 1$ and $0 < \mu < 1$.

Two traffic scenarios are used to evaluate system performance:

- Bernoulli (uncorrelated) arrival: At an input port *i*, in each time slot, the probability that a new packet arrives is equal to λ_i, which is independent of any other time slot.
- Bursty (correlated) arrival: At each input link, *busy* burst and *idle* burst occur in turn. A *busy* burst is a sequence of consecutive *busy* time slots (with packets arriving). An *idle* burst is a sequence of consecutive *idle* time slots (without a packet arriving). As a random variable, the length of a *busy* burst, *b*, follows geometrical distribution with a mean of $E[b]$; the length of an *idle* burst follows geometrical distribution with a mean of $(1 / \lambda_i - 1) \times E[b]$ at an input port *i*.

5.2 Integrated Scheduling within One Switching Fabric

The integrated scheduling algorithm logically includes unicast scheduling, multicast scheduling, and the integration strategy.

5.2.1 Unicast Scheduling and Multicast Scheduling

With the proposed integrated scheduling algorithms, unicast and multicast scheduling utilize the advances in the research of unicast and multicast scheduling algorithms, respectively. In order to achieve a good performance, there are two standards in selecting unicast and multicast scheduling algorithms. Firstly, both unicast and multicast scheduling algorithms need to achieve an effective performance in their own scheduling domain. Secondly, the selected unicast and multicast scheduling algorithms can be integrated smoothly. The well-known unicast scheduling algorithm *i*SLIP [32] achieves 100% throughput and exhibits a solid delay performance for uniform Bernoulli and bursty unicast traffic. Therefore, *i*SLIP is chosen as the base of unicast scheduling. Unicast scheduling consists of multiple iterations, each of which includes three steps:

Step 1: Unicast request. Each unicast VOQ at each unmatched input port sends a request to the corresponding output port if there is a HOL packet.

Step 2: Unicast grant. At each unmatched output port, if one or more requests are received, the one originated from the input port that is closest to the highest priority pointer of a round-robin schedule is granted and a grant is sent to the corresponding input port. Its highest priority pointer is moved to one location beyond the granted input port if and only if the grant is accepted by that input port in the next step of the *first iteration*.

Step 3: Unicast accept. At each unmatched input port, if one or more grants are received, the one issued by the output port that is closest to the highest priority pointer of a round-robin schedule is accepted and the corresponding output port is notified. The highest priority pointer is moved to one location beyond the accepted output port. This input port is matched with the accepted output port.

Regarding multicast scheduling, WBA performs well and is simple to implement in hardware [44]. However, WBA only works for switches with one multicast queue and therefore suffers HOL blocking. A variation of WBA, *weight based algorithm for multiple multicast queues* (WBA-MQ), is adopted as the base of multicast scheduling. Multicast scheduling consists of multiple iterations, each of which has three steps:

Step 1: Multicast request. At each unmatched input port, the weight of the HOL packet in each non-empty multicast queue is sent together with multicast request to all its destined output ports.

Step 2: Multicast grant. At each unmatched output port, if one or more requests are received, the one with the maximal weight is granted and a grant is sent to the corresponding input port. Ties are solved randomly.

Step 3: Multicast accept. At each unmatched input port, if one or more grants are received, all the grants belonging to the HOL packet with the highest *granting percentage* are accepted. The granting percentage is a ratio of the number of grants to the remaining fanout size. This input port is matched with all the output ports that issue the accepted grants.

5.2.2 Loosely and Fully Slot-Coupled Integration Algorithms: lSCIA and fSCIA

Briefly speaking, the integrated scheduling procedure proposed in this chapter works as follows:

1) Decide the multicast service ratio (S_m), which is indicated by the probability that a time slot is identified to schedule multicast traffic first. Such a time slot is called a *multicast slot*. A time slot identified to schedule unicast traffic first is called a *unicast slot*.

2) At the beginning of each time slot, tag the time slot as either a multicast slot or unicast slot with the following probabilities:

$$\Pr\{\text{unicast slot}\} = 1 - S_m,$$

$$\Pr\{\text{multicast slot}\} = S_m.$$

3) In a unicast (multicast) slot, unicast scheduling and multicast scheduling are coordinated together while unicast (multicast) scheduling has higher priority.

```
Tag current slot.
done = –1;
if current slot is a unicast slot, then
  while (there is unmatched input/output ports) and (done != 0)
    Unicast request
    Unicast grant
    Unicast accept
    done = number of matches in this iteration
  end
  done = –1;
  while (there is unmatched input/output ports) and (done != 0)
    Multicast request
    Multicast grant
    Multicast accept
    done = number of matches in this iteration
  end
else
  while (there is unmatched input/output ports) and (done != 0)
    Multiast request
    Multicast grant
    Multicast accept
    done = number of matches in this iteration
  end
  done = –1;
  while (there is unmatched input/output ports) and (done != 0)
    Unicast request
    Unicast grant
    Unicast accept
    done = number of matches in this iteration
  end
end
```

Figure 29. The pseudo codes of *l*SCIA.

Assuming that the traffic pattern is stable for a switch during a specific period, S_m can be determined before scheduling. The value of S_m is chosen within [0, 1] where different values result in different performance. The selection of S_m and its impact on the performance are going to be discussed in Section 5.3. Here, we assume that S_m has been properly chosen.

The way to implement the coordination between unicast and multicast scheduling is called integration strategy. One intuitive way is the loosely coupled integration strategy. In unicast (multicast) slots, firstly finish all unicast (multicast) iterations; and secondly perform multicast (unicast) iterations to use the unmatched input ports and output ports. Based on this strategy, we propose *l*SCIA. Its detail procedures are shown in Figure 29.

Considering that both unicast and multicast scheduling consist of multiple iterations and both unicast and multicast iterations consist of three steps: *request, grant*, and *accept*, an alternative way to implement the third step is the fully coupled integration strategy. During each iteration, unicast and multicast requests, grants, and accepts are performed together. Based on this strategy, we propose *f*SCIA, which includes the following steps:

Step 1: Tagging slot. Tag current time slot as either a unicast slot or a multicast slot randomly based on S_m.

Step 2: Request. At each unmatched input port, every non-empty unicast VOQ sends a unicast request to the corresponding output port. Each non-empty multicast queue sends the weight of HOL packet together with multicast requests to all destined output ports of that HOL packet.

Step 3: Grant. At each unmatched output port, in a unicast (multicast) slot, one unicast (multicast) request is granted with higher priority. One multicast (unicast) request will be granted only if there is no unicast (multicast) request. If a unicast (multicast) request is granted, a unicast (multicast) grant is sent to the corresponding input port.

Step 4: Accept. At each unmatched input port, in a unicast (multicast) slot, unicast (multicast) grants belonging to a queue are accepted with higher priority. Multicast (unicast) grants are accepted only if there is no unicast (multicast) request. Input port is matched with all the output ports that issue the accepted grants.

Step 5: Iterate *step 2 ~ step 4* until there is no unmatched input/output ports, or the number of matches during the last iteration is zero.

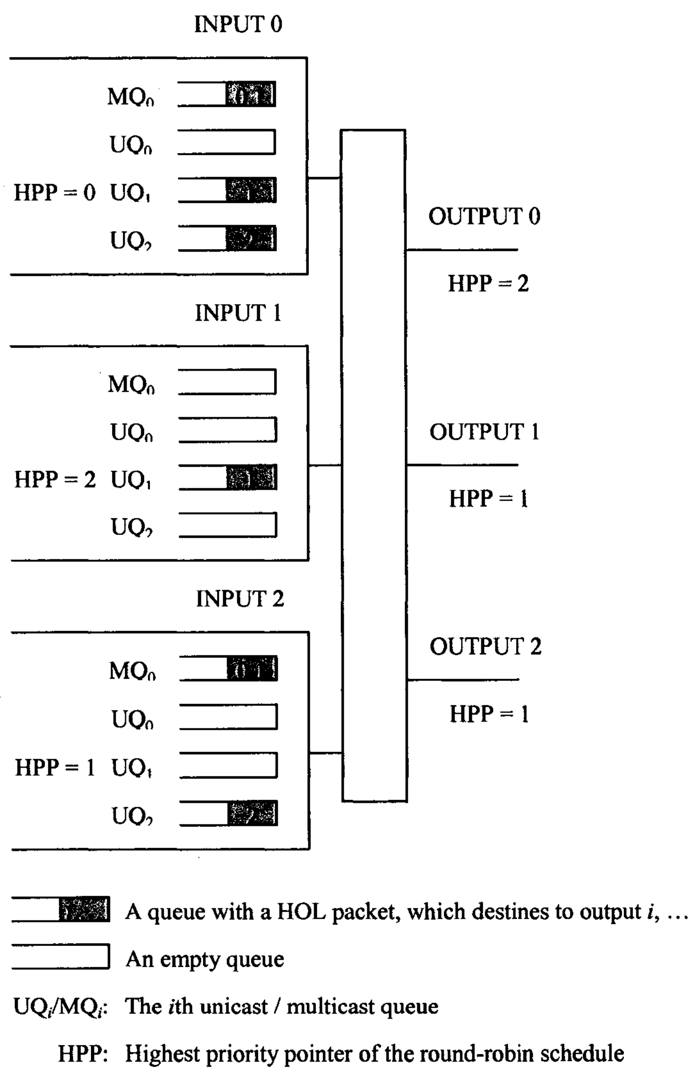

INPUT 0
MQ0
UQ0
HPP = 0 UQ1
UQ2
INPUT 1
MQ0
UQ0
HPP = 2 UQ1
UQ2
INPUT 2
MQ0
UQ0
HPP = 1 UQ1
UQ2
OUTPUT 0
HPP = 2
OUTPUT 1
HPP = 1
OUTPUT 2
HPP = 1
A queue with a HOL packet, which destines to output i, ...
An empty queue
UQi/MQi: The ith unicast / multicast queue
HPP: Highest priority pointer of the round-robin schedule

Figure 30. A snapshot of a 3×3 switch at the beginning of a time slot.

Table 5. Matches that are set up during a multicast slot with *I*SCIA.

Iterations	Matched Input	Matched Queue	Matched Output
1st	0	MQ_0	0
	0	MQ_0	1
2nd	2	UQ_2	2

Table 6. Matches that are set up during a unicast slot with *I*SCIA.

Iterations	Matched Input	Matched Queue	Matched Output
1st	1	UQ_1	1
	2	UQ_2	2
2nd	0	MQ_0	0

Table 7. Matches that are set up during a multicast slot with *f*SCIA.

Iterations	Matched Input	Matched Queue	Matched Output
1st	0	MQ_0	0
	0	MQ_0	1
	2	UQ_2	2

Table 8. Matches that are set up during a unicast slot with *f*SCIA.

Iterations	Matched Input	Matched Queue	Matched Output
1st	0	MQ_0	0
	1	UQ_1	1
	2	UQ_2	2

An example of the matches set up by running *l*SCIA and *f*SCIA is given as following. A snapshot of a 3×3 switch at the beginning of a time slot is shown in Figure 30. At each input, there are one multicast queue and three unicast queues. Assuming that the HOL packet at MQ_0 of input 0 has a higher weight than the HOL packet at MQ_0 of input 2, the matches that are set up during a multicast (unicast) slot with *l*SCIA (*f*SCIA) are shown in Tables 5, 6, 7, and 8, respectively.

5.2.3 Properties of lSCIA and fSCIA

In this subsection, we discuss some desirable properties of two slot-coupled integration algorithms, proposed in the previous subsection.

Property 1: Maximal match. A set of maximal matches between input ports and output ports is found, i.e., no more matches can be made without removing existed matches belonging to that set.

Property 2: Distributed scheduling. Both of them are distributed scheduling algorithms. Each input/output arbitrates independently of other inputs/outputs. So arbitrations can be made at different inputs/outputs in parallel. With *l*SCIA, in a unicast iteration, since an output/input receives at most *N* unicast requests/grants, the complexity of unicast granting/accepting is O(*N*). Therefore, the complexity of a unicast iteration is O(*N*). With *l*SCIA, in a multicast iteration, since an output receives at most *K* multicast requests and an input needs to select a multicast HOL packet among at most *K* multicast

HOL packets, the complexity of multicast granting/accepting is O(K). Therefore, the complexity of a multicast iteration is O(K). With *f*SCIA, in an iteration, an output receives at most ($N + K$) requests and an input needs to select one HOL packet among at most ($N + K$) HOL packets, the complexity of granting/accepting is O($N + K$). Therefore, the complexity of an iteration is O($N + K$). Requests, grants and accepts can be performed in parallel so that the scheduling overhead is much less than the centralized scheduler.

Property 3: Convergence. Both of them converge in at most N iterations. Because both unicast iteration and multicast iteration will stop when the number of matches in the last iteration is zero or there is no unmatched output port, the slowest convergence procedure is setting up one match per iteration. Considering that at most N matches will be set up, the number of iterations will be no bigger than N. Specially, with *f*SCIA, unicast and multicast requests, grants, and accepts are integrated at the level of iteration and performed in parallel. Therefore, *f*SCIA is expected to converge faster than *l*SCIA.

Through extensive simulation study of various traffic patterns and switches with different sizes, we observed that (1) the average number of iterations in *f*SCIA is no bigger than $\log_2(N)$, and the average number of iterations in *l*SCIA is no bigger than $2\times\log_2(N)$; (2) *f*SCIA always converges faster than *l*SCIA. For instance, with *f*SCIA and *l*SCIA, we show the simulation results for a 16×16 switch under Bernoulli traffic and bursty traffic with $E[b] = 32$, respectively. With output load varying within (0, 1), the average numbers of iterations are shown in Figure 31, where multicast output load is fixed as 69.5% of the output load. For the given output load (0.9 for Bernoulli traffic and 0.8 for bursty traffic), with μ_m/μ varying from 0.1 to 0.9, the average number of iterations are shown in Figure 32. We can see that under different traffic loads and different traffic compositions, *f*SCIA always converges approximately 50% faster than *l*SCIA.

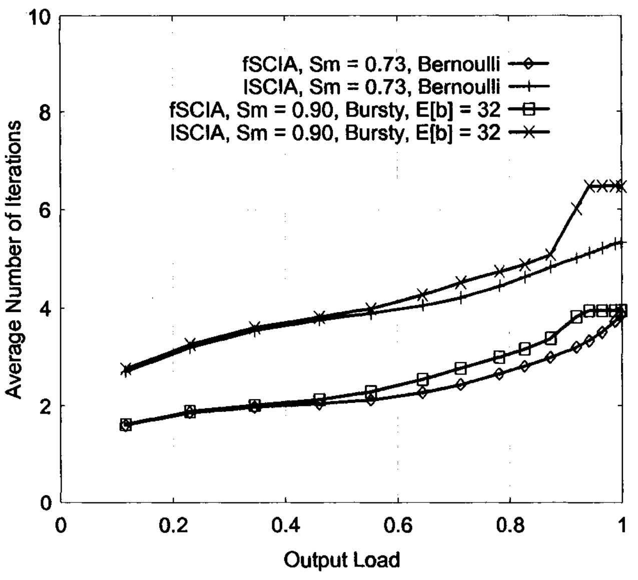


Figure 31. Average number of iterations as a function of normalized output load with uniform traffic, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $K = 4$.

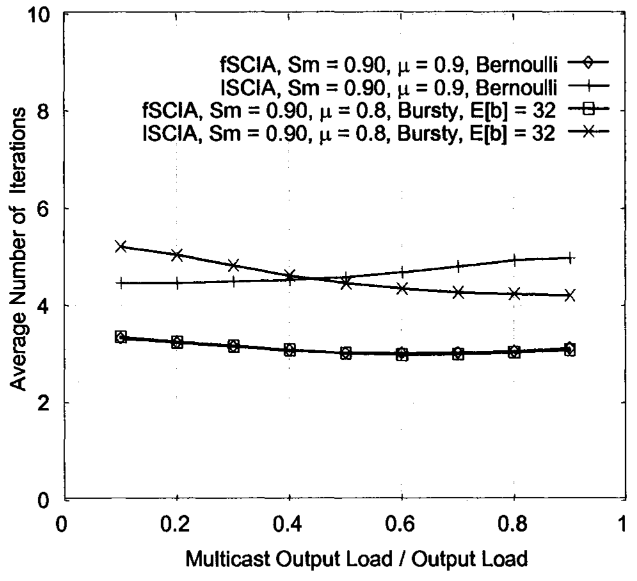


Figure 32. Average number of iterations as a function of μ_m/μ with uniform traffic, $F_{min} = 2$, $F_{max} = 6$, and $K = 16$.

Property 4: Fair resource allocation. Unlike the scheme proposed in [2], which always gives higher priority to multicast scheduling, a parameter, S_m, is provided in the design to adjust the allocation of scheduling priority between unicast and multicast traffic. The larger the value of S_m, the more switching fabric resource is allocated to multicast scheduling. The switching fabric resource allocation between unicast and multicast traffic can be adjusted by adjusting the value of S_m. Considering that μ_m/μ represents the percentage of multicast traffic in terms of output packets, in *l*SCIA and *f*SCIA, the closer to μ_m/μ the value of S_m, the more fair. However, simply let S_m equal to μ_m/μ, 100% throughput cannot be assured. The selection of S_m is further discussed in Section 5.3.

5.2.4 Algorithms for Performance Comparison

As what was discussed in Section 2.3, three integration schemes based on sharing one switching fabric have been proposed. The integration schemes proposed in [2] and [25] cannot achieve 100% throughput. With the scheme proposed in [2], the saturation throughput linearly decreases from 1 to 0.5 as μ_m increases from 0 to 0.4. When μ_m is bigger than 0.4, switch becomes unstable because multicast queues become unstable. With the scheme proposed in [25], under the Bernoulli traffic assumed by the authors, the saturation throughput is less than 0.8. In the following subsections, it is going to be shown that both *l*SCIA and *f*SCIA can achieve 100% throughput under different traffic compositions with various percentages of multicast traffic. Therefore, the performances of *l*SCIA and *f*SCIA are obviously better than those two schemes. By logically splitting a multicast packet into unicast copies and storing them in the corresponding virtual output queue at each input port, MSM [38] can achieve 100% throughput. Then, from the saturation throughput point of view, *l*SCIA and *f*SCIA perform the same as MSM. In the following subsections, it is going to be shown that *l*SCIA and *f*SCIA also perform better than MSM in terms of delay and packet loss ratio.

Besides MSM, a random integration algorithm (RIA) is introduced for performance comparison. RIA also consists of request, grant and accept. At each unmatched input port, every non-empty unicast VOQ sends a *unicast request* to the corresponding output port, and every non-empty multicast queue sends *multicast requests* to all destined output ports of the HOL packet. At each unmatched output port, if there are one or more requests, one of them is granted randomly and a grant is sent to the corresponding input port. At each unmatched input port, the received grants are associated with the corresponding unicast virtual output queues and multicast queues, respectively. One of those queues that have one or more than one grants is selected randomly. The grants belonging to the selected queue are accepted and this input port is matched with the corresponding output ports that issue those accepted grants. The above procedure iterates until there are no unmatched output/input ports, or the number of matches during the last iteration is zero. RIA is expected to have worse performance compared with *l*SCIA and *f*SCIA. By comparing their simulated performance, the performance improvement introduced by adopting *i*SLIP for unicast scheduling and WBA-MQ for multicast scheduling is illustrated.

In addition, in order to compare the performance of *l*SCIA and *f*SCIA with the scheme that separates unicast and multicast traffic in different fabrics, non-coupled integration algorithm (NCIA) is introduced. NCIA is a time-division variation of the integration approach using isolated switching fabric. One switching fabric is shared by unicast traffic and multicast traffic based on the traffic composition, i.e., $S_m = \mu_m / \mu$. In unicast (multicast) slots, the switching fabric is used to transmit unicast (multicast) packets only. At the beginning of each time slot, tag the current time slot as either a *unicast slot* or a *multicast slot* randomly. In a unicast (multicast) slot, perform iterations of unicast (multicast) request, unicast (multicast) grant, and unicast (multicast) accept repeatedly, until there is no unmatched output/input or the number of matches during the last iteration is zero. With the scheme that separates unicast and multicast traffic in different time slots, some outputs may be idle, since in unicast (multicast) slots, multicast (unicast) packets are not transmitted, even if there are unmatched inputs and outputs after unicast (multicast) iterations. Its performance is expected to be worse than *l*SCIA and *f*SCIA. Through simulations, in Section 5.4, the performance improvement gained by coupling unicast and multicast scheduling in each time slot is demonstrated.

5.3 The Multicast Service Ratio in a Time Slot

In this subsection, the working interval of S_m is analyzed and the selection of S_m is discussed.

5.3.1 Working Interval of S_m

For a given traffic pattern, switching performance varies as the value of S_m varies from one to zero. Since multicast scheduling cannot achieve 100% throughput, the switching system may become unstable when S_m belongs to some interval, even if the incoming traffic is not overloaded. For a specific traffic pattern, the *working interval* of S_m is defined as a subset of [0, 1] such that the saturation throughput is no less than the output load when the value of S_m falls into this interval.

Before proceeding, we define the following notations:

- M_u, M_m (M_u', M_m'): The unicast matching rate and the multicast matching rate in unicast (multicast) slots. The matching rate is defined as the number of unicast or multicast matches over the number of output ports, *N*.
- $\mu_{m\text{-}sat}$: The saturation throughput of a multicast scheduling algorithm for pure multicast traffic (P_u equals to 0).

Under the assumption of uniform input traffic, when the throughput of the switching system is not saturated, we have

$$(1-S_m)\times E[M_u]+S_m\times E[M_u']=\lambda_u, \tag{68}$$

$$(1-S_m)\times E[M_m]+S_m\times E[M_m']=\lambda_m\times E[F]. \tag{69}$$

Also, due to the assumption that the speedup of switching fabric equals to one, the four matching rates must follow the following inequalities.

$$E[M_u]\le 1, \tag{70}$$

$$E[M_m]\le \mu_{m-sat}\times(1-E[M_u]), \tag{71}$$

$$E[M_m']\le \mu_{m-sat}, \tag{72}$$

$$E[M_u']\le 1-E[M_m']. \tag{73}$$

Recall that traffic is admissible. Thus,

$$\lambda_u+\lambda_m\times E[F]<1. \tag{74}$$

In multicast slots, after finishing the multicast traffic scheduling, especially if multicast scheduling is concentrative [44], the number of unmatched input ports is more than the number of unmatched output ports. By employing VOQ, the remaining unmatched output ports could be fully utilized by unicast scheduling. The throughput capacity of switching fabric indicates the maximum throughput that can be achieved for a given type of traffic. Considering that unicast scheduling algorithm, which is *i*SLIP in this paper, can achieve 100% throughput, the throughput capacity of switching fabric for unicast traffic in a multicast slot (ξ_u^m) is given by

$$\xi_u^m = (1 - E[M_m^{'}]). \tag{75}$$

Similarly, in unicast slots, the throughput capacity of switching fabric for unicast traffic in a unicast slot (ξ_u^u) is given by

$$\xi_u^u = 1. \tag{76}$$

Therefore, as to the throughput capacity of switching fabric for unicast traffic (ξ_u),

$$\xi_u = (1 - S_m) \times \xi_u^u + S_m \times \xi_u^m = 1 - S_m + S_m \times (1 - E[M_m^{'}]). \tag{77}$$

Combining (68), (70), (73), and (77), we have $\lambda_u \leq \xi_u$. This means that the incoming unicast traffic can always be transmitted within a bounded number of time slots for any $S_m \in [0,1]$. Put another way, the switching system becomes unstable only when the multicast traffic load exceeds the throughput capacity of switching fabric for multicast traffic (ξ_m).

Due to the HOL blocking in multicast scheduling, the throughput capacity of switching fabric for multicast traffic in a multicast slot (ξ_m^m) is given by

$$\xi_m^m = \mu_{m-sat}. \tag{78}$$

In unicast slots, although the number of unmatched input ports equals to the number of unmatched output ports, the random occupancy of output ports by unicast scheduling decreases the concentration of multicast scheduling. Accordingly, the throughput capacity of switching fabric for multicast traffic in a unicast slot (ξ_m^u) follows the following relationship:

$$\xi_m^u < \mu_{m-sat} \times (1 - E[M_u]). \tag{79}$$

Considering that $\xi_m = S_m \times \xi_m^m + (1 - S_m) \times \xi_m^u$, (78), and (79), we have

$$\xi_m \leq S_m \times \mu_{m-sat} + (1 - S_m) \times \mu_{m-sat} \times (1 - E[M_u]), \tag{80}$$

$$\xi_m \geq S_m \times \mu_{m-sat}. \tag{81}$$

Clearly, $\xi_m^m \geq \xi_m^u$. Thus, ξ_m is a non-decreasing function of S_m and it achieves its maximal value when S_m equals to 1. During the decrement of ξ_m, the switching system is stable until the multicast output load exceeds ξ_m. Consequently, the working interval of S_m could be expressed as $(S_{m\_min}, 1]$, where S_{m_min} is determined by several factors, for instance, traffic pattern, multicast scheduling algorithm, and the interaction between unicast scheduling and multicast scheduling. Next, it is going to derive the upper bound and lower bound of S_{m_min}.

From (69) and (71), we have

$$\frac{\lambda_m \times E[F] - S_m \times E[M_m']}{(1-S_m)} \le \mu_{m-sat} \times (1-E[M_u]). \tag{82}$$

Similarly from (68) and (73), we have

$$1-E[M_u] \le \frac{1-\lambda_u - S_m \times E[M_m']}{1-S_m}. \tag{83}$$

Considering that $\mu_{m-sat} \in (0,1)$, (72), (82) and (83), we have

$$S_m \ge \frac{\lambda_m \times E[F] - \mu_{m-sat} \times (1-\lambda_u)}{\mu_{m-sat} \times (1-\mu_{m-sat})}. \tag{84}$$

Therefore, the lower bound of S_{m_min}, $S_{m\_min\_low}$, is given by

$$S_{m\_min\_low} = \begin{cases} 0, & \dfrac{\lambda_m \times E[F]}{\mu_{m-sat}} \le (1-\lambda_u); \\ \dfrac{\lambda_m \times E[F] - \mu_{m-sat} \times (1-\lambda_u)}{\mu_{m-sat} \times (1-\mu_{m-sat})}, & \text{otherwise.} \end{cases} \tag{85}$$

As what was mentioned previously, the switch is stable if multicast output load is no bigger than the throughput capacity of switching fabric for multicast traffic, i.e.,

$$\lambda_m \times E[F] \le \xi_m. \tag{86}$$

Considering (81), the switch must be stable when

$$\lambda_m \times E[F] \le S_m \times \mu_{m-sat}. \tag{87}$$

Therefore, the upper bound of S_{m_min}, $S_{m\_min\_up}$, is

$$S_{m\_min\_up} = \begin{cases} 1, & \lambda_m \times E[F] \ge \mu_{m-sat}; \\ \dfrac{\lambda_m \times E[F]}{\mu_{m-sat}}, & \text{otherwise.} \end{cases} \tag{88}$$

In conclusion, as to the working interval of S_m, we have

$$(S_{m\_min\_low}, 1] \supseteq (S_{m\_min}, 1] \supseteq (S_{m\_min\_up}, 1]. \tag{89}$$

where $(S_{m\_min\_up}, 1]$ is named as the working interval's upper bound, and $(S_{m\_min\_low}, 1]$ is named as the working interval's lower bound.

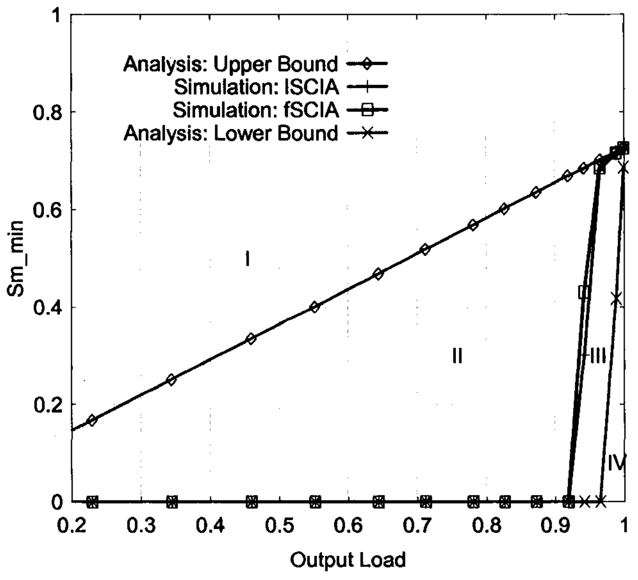


Figure 33. S_{m_min} as a function of normalized output load with uniform Bernoulli traffic, P_u = 70%, F_{min} = 2, F_{max} = 6, and K = 4.

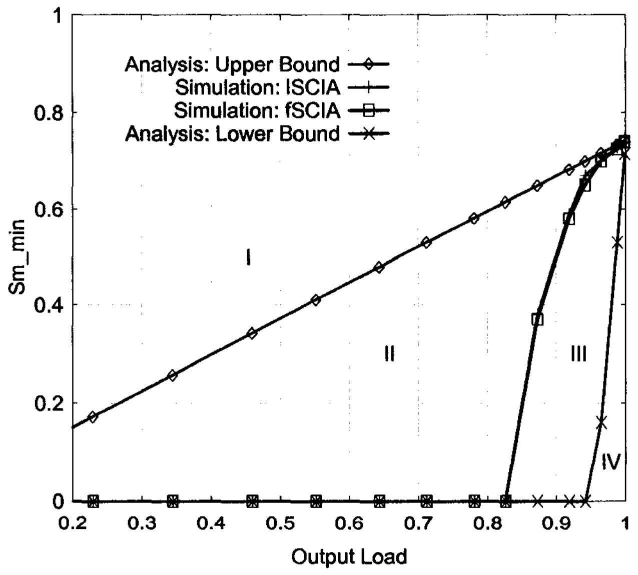


Figure 34. S_{m_min} as a function of normalized output load with uniform bursty traffic, $E[b]$ = 32, P_u = 70%, F_{min} = 2, F_{max} = 6, and K = 16.

The above analysis on working interval's upper and lower bounds only applies to uniform traffic, because *i*SLIP is assured to achieve 100% throughput only for uniform traffic. Both *i*SLIP and WBA are designed for uniform traffic and their performances are evaluated under the assumption of uniform traffic in [32] and [44], respectively. Therefore, *l*SCIA and *f*SCIA are mainly proposed for uniform traffic and 100% throughput is theoretically guaranteed only for uniform traffic. However, the idea of coupling unicast and multicast scheduling together in one time slot could be extended to design integration algorithms specifically for non-uniform traffic. In addition, for non-uniform traffic, the performances of *l*SCIA and *f*SCIA are evaluated through simulations and the results are given in Section 5.4.

For a given traffic pattern, by running simulations with S_m varying within [0, 1], S_{m_min} can be identified. The simulation results for a 16 × 16 switch are collected. Figures 33 and 34 plot the theoretical upper and lower bounds of S_{m_min}, and the simulation values of S_{m_min} for Bernoulli traffic and bursty traffic, respectively. The simulation values of S_{m_min} for *l*SCIA and *f*SCIA are almost overlapped with each other. If S_m falls into areas I and II, switch is stable; if S_m falls into areas III and IV, switch is unstable. We can observe that the simulated value of S_{m_min} is very close to the theoretical value of $S_{m\_min\_up}$ when output load equals to 1.

5.3.2 Selection of S_m

According to the aforementioned analysis, $S_{m\_min\_up}$ increases as λ increases for a given traffic pattern, which includes the traffic scenario (Bernoulli or bursty), $E[b]$, P_u, F_{min}, and F_{max}. Recall that we only consider admissible traffic, μ equals to 1 when λ reaches the maximal value, λ_{max}. For a given traffic pattern, the intersection of working intervals' upper bounds under different values of λ equals to the working interval's upper bound when λ equals to λ_{max}, i.e., $(S_{m\_min\_up}(\lambda_{max}), 1]$. As long as the selected S_m falls within that intersection, with theoretical guarantee, the switch can achieve 100% throughput. Two examples of such an intersection as a function of μ_m/μ are given in Tables 9 and 10. 100% throughput can be achieved for different traffic compositions with the percentage of multicast traffic varying from 0.1 to 0.9 through selecting an S_m within the corresponding intersection.

Table 9. Intersection of working intervals' upper bounds as a function of μ_m/μ with uniform Bernoulli traffic, $F_{min} = 2$, $F_{max} = 6$, $K = 4$, $\mu_{m\text{-}sat} = 0.956$, $N = 16$.

μ_m/μ	0.1	0.3	0.5	0.7	0.9
Intersection	(0.11,1]	(0.32,1]	(0.53,1]	(0.73,1]	(0.94,1]

Table 10. Intersection of working intervals' upper bounds as a function of μ_m/μ with uniform bursty traffic, $E[b] = 32$ $F_{min} = 2$, $F_{max} = 6$, $K = 16$, $\mu_{m\text{-}sat} = 0.935$, $N = 16$.

μ_m/μ	0.1	0.3	0.5	0.7	0.9
Intersection	(0.11,1]	(0.32,1]	(0.54,1]	(0.75,1]	(0.96,1]

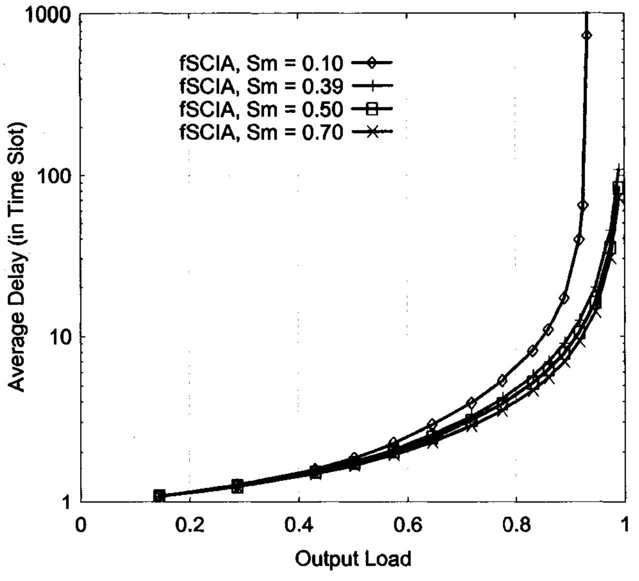
It is clear that the bigger the value of S_m, the better the performance of multicast scheduling, and the worse the performance of unicast scheduling. In order to achieve fair resource allocation, S_m should be as close to μ_m/μ, which equals to $(1-P_u)\times E[F]/(P_u+(1-P_u)\times E[F])$, as possible.

From (88) and considering that $\mu = 1$ when $\lambda = \lambda_{max}$, we have

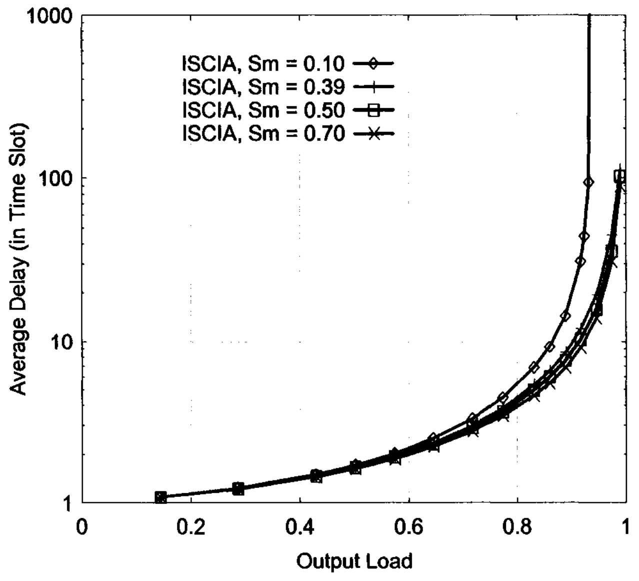
$$S_{m\_min\_up}(\lambda_{\max}) = \begin{cases} 1, & \mu_m/\mu \geq \mu_{m-sat}; \\ \dfrac{\mu_m}{\mu\times\mu_{m-sat}}, & \text{otherwise.} \end{cases} \tag{90}$$

As we know, $\mu_{m\text{-}sat} \leq 1$. Then, $\mu_m/\mu \notin (S_{m\_min\_up}(\lambda_{max}), 1]$. Simply let S_m be μ_m/μ cannot assure the switch achieves 100% throughput. Instead, S_m should be a value a little bigger than $S_{m\_min\_up}(\lambda_{max})$ to assure both 100% throughput and fair resource allocation.

Through running extensive simulations under different traffic compositions, as we expected, both *l*SCIA and *f*SCIA achieve 100% throughput and bounded delay performance with S_m within the intersection of working intervals' upper bounds, while 100% throughputs cannot be achieved with S_m outside the union of working intervals' lower bounds. Figures 35 and 36 show the average delay as a function of output load under different values of S_m for Bernoulli and bursty traffic, respectively. Increasing S_m from a value outside working interval to a value inside working interval improves the delay performance significantly. However, the difference between the average delays under two different S_ms within working intervals is very small, although the average delay does decrease as S_m increases.

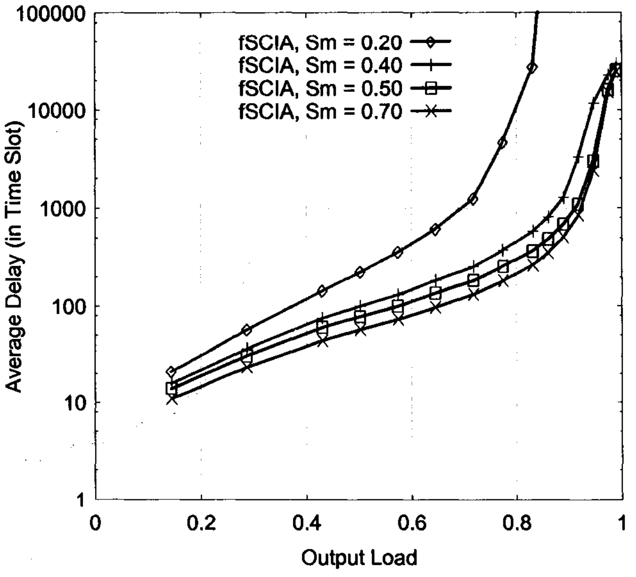


(a) *f*SCIA

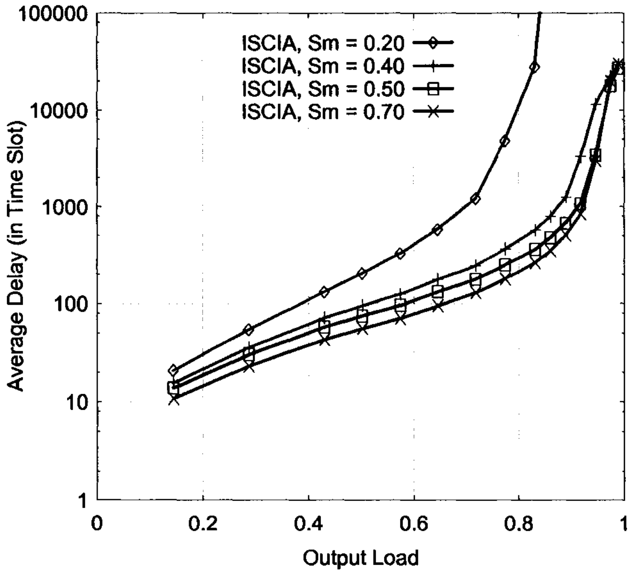


(b) *l*SCIA

Figure 35. Delay as a function of normalized output load with uniform Bernoulli traffic, $P_u = 90\%$, $\mu_m/\mu = 37\%$, $F_{min} = 2$, $F_{max} = 6$, $K = 4$, intersection of working intervals' upper bounds being (0.388, 1], and union of working intervals' lower bounds being (0.107, 1].



(a) *f*SCIA



(b) *l*SCIA

Figure 36. Delay as a function of normalized output load with uniform bursty traffic, $E[b] = 32$, $P_u = 90\%$, $\mu_m/\mu = 37\%$, $F_{min} = 2$, $F_{max} = 6$, $K = 16$, intersection of working intervals' upper bounds is (0.392, 1], union of working intervals' lower bounds is (0.205, 1].

5.3.3 The Impact of $\mu_{m\text{-}sat}$ to $S_{m\_min\_up}$

For a given traffic pattern and a given λ, $S_{m\_min\_up}$ increases as $\mu_{m\text{-}sat}$ increases, which means the working interval can be enlarged by improving the saturation throughput of multicast scheduling. From (90), $S_{m\_min\_up}(\lambda_{max})$ is a non-decreasing function of $\mu_{m\text{-}sat}$ and $S_{m\_min\_up}(\lambda_{max})$ equals to μ_m/μ when the value of $\mu_{m\text{-}sat}$ reaches 1. Thus, as long as $\mu_{m\text{-}sat}$ is big enough, a value that is close to μ_m/μ can be selected within the intersection of working intervals' upper bounds.

The value of $\mu_{m\text{-}sat}$ is determined by traffic scenario (Bernoulli or bursty), $E[b]$, F_{min}, F_{max}, and K. Values of $\mu_{m\text{-}sat}$ are gained for given traffic scenarios and K through simulations. Tables 11 and 12 demonstrate the values of $\mu_{m\text{-}sat}$ as a function of K under uniform Bernoulli and bursty traffic, respectively. It is shown that $\mu_{m\text{-}sat}$ can be enlarged to approach 1 by increasing the value of K.

Table 11. $\mu_{m\text{-}sat}$ as a function of K with uniform Bernoulli traffic, $F_{min} = 2$, $F_{max} = 6$, and $N = 16$.

k	1	4	16	32	64
$\mu_{m\text{-}sat}$	0.873	0.956	0.987	0.993	0.997

Table 12. $\mu_{m\text{-}sat}$ as a function of K with uniform bursty traffic, $E[b] = 32$, $F_{min} = 2$, $F_{max} = 6$, and $N = 16$.

k	1	4	16	32	64
$\mu_{m\text{-}sat}$	0.721	0.835	0.935	0.964	0.977

5.4 Analytical and Simulation Results

Extensive simulations are performed for various traffic patterns, and switches with different sizes to verify the theoretical analysis and infer further conclusions. The simulation results are consistent with the theoretical analysis results. Because of the space limitation, only some typical results of a 16 × 16 input-queued switch are shown. All simulations have been fixed at one million time slots. Data are gathered for statistical elaboration during the last half million time slots. Unless specified otherwise, the input buffer size (L) is 5,000 packets for Bernoulli traffic and 100,000 packets for bursty traffic. This is to assure that the packet loss ratio is zero while evaluating the throughput and delay performance of algorithms that can achieve 100% throughput.

5.4.1 Performance of Integration Algorithms

5.4.1.1 Uniform Traffic

Figures 37 and 38 show the average delays as a function of output load for *f*SCIA, *l*SCIA, NCIA, RIA, and MSM, under Bernoulli traffic and bursty traffic with $E[b] = 32$, respectively. By assigning S_m a value within the intersection of working intervals' upper bounds, 100% throughputs are achieved by *f*SCIA and *l*SCIA with theoretical guarantee. On the contrary, with RIA or NCIA, the switch becomes unstable once output load exceeds a certain value less than 1, i.e., their saturation throughput is less than 1. Furthermore, average delays of *f*SCIA and *l*SCIA are always smaller than RIA and NCIA. In a word, in terms of throughput and delay, *f*SCIA and *l*SCIA perform better than RIA and NCIA. The performance improvement introduced by adopting *i*SLIP as the base for unicast scheduling and adopting WBA-MQ as the base for multicast scheduling is shown by the delay and saturation throughput differences between RIA and *f*SCIA (*l*SCIA). Similarly, compared with NCIA, the promising throughput and delay performance of *f*SCIA and *l*SCIA demonstrates the benefits of integrating unicast and multicast scheduling in one switching fabric.

As shown in [38], MSM can also achieve 100% throughput. From the saturation throughput point of view, MSM performs as well as *f*SCIA and *l*SCIA. However, under both Bernoulli and bursty traffic, the average delays of two slot-coupled integration algorithms are obviously smaller than the average delay of MSM no matter what the output load is. *f*SCIA and *l*SCIA exhibit better delay performance than MSM. This is not surprising. With MSM, a multicast packet is logically split into unicast copies and scheduled individually. Then the scheduling of multicast traffic is not concentrative. According to [44], this kind of scheme compromises the performance of multicast scheduling. However, with the slot-coupled integration algorithms, through queueing and scheduling unicast and multicast traffic separately, the merits of well-studied unicast and multicast scheduling algorithms and queueing policies could be captured so that both unicast and multicast scheduling can achieve good performances.

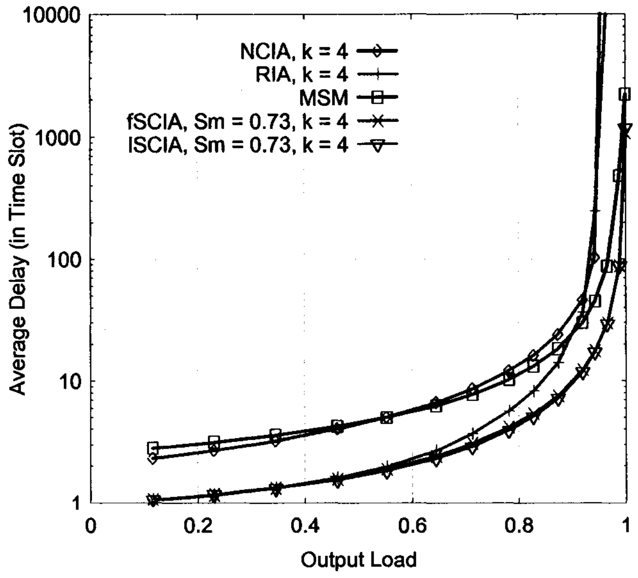


Figure 37. Delay as a function of normalized output load with uniform Bernoulli traffic, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, and $F_{max} = 6$.

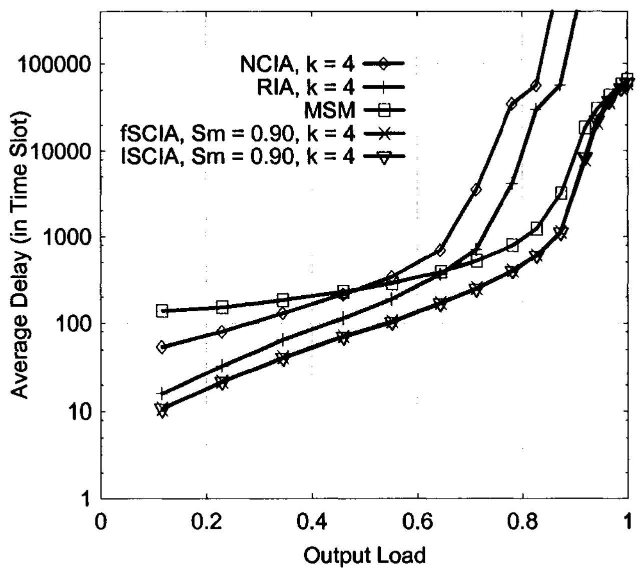


Figure 38. Delay as a function of normalized output load with uniform Bursty traffic, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, and $F_{max} = 6$.

From the packet loss ratio point of view, *f*SCIA and *l*SCIA also perform better than MSM, RIA and NCIA. The average packet loss ratios as a function of output load are illustrated and compared in Figures 39 and 40. The input buffer size is fixed at 100 packets for Bernoulli traffic and 1000 packets for bursty traffic, respectively. The advantage of *f*SCIA and *l*SCIA in terms of packet loss ratio is obvious, which is consistent with the comparison result of saturation throughput and delay.

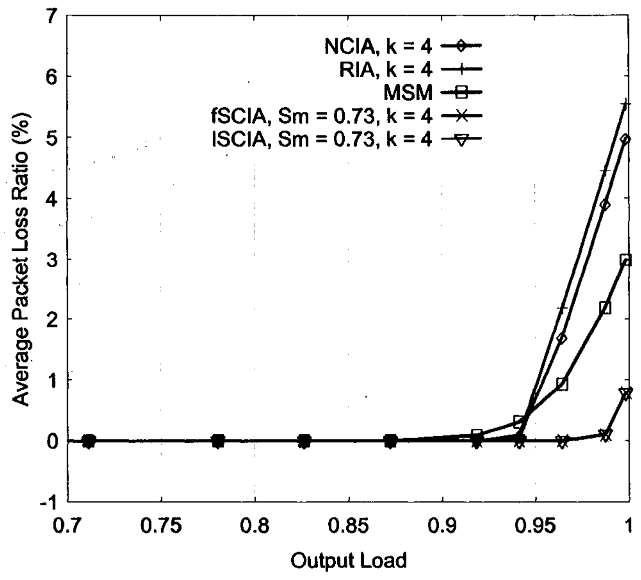


Figure 39. Packet loss ratio as a function of normalized output load with uniform Bernoulli traffic, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $L = 100$.

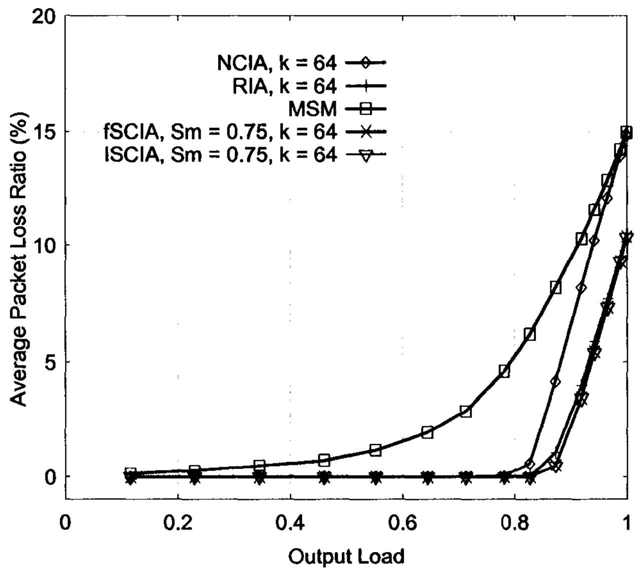


Figure 40. Packet loss ratio as a function of normalized output load with uniform bursty traffic, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $L = 1000$.

Additionally, two groups of simulations are run with two given output loads, respectively, where the percentage of multicast traffic varies from 0.1 to 0.9. The average delays under Bernoulli traffic and bursty traffic are shown in Figures 41 and 42, respectively. We can observe that the average delays of *l*SCIA and *f*SCIA are always smaller than that of MSM, RIA, and NCIA. This result further illustrates the performance advantage of the slot-coupled integration algorithms under variant traffic compositions.

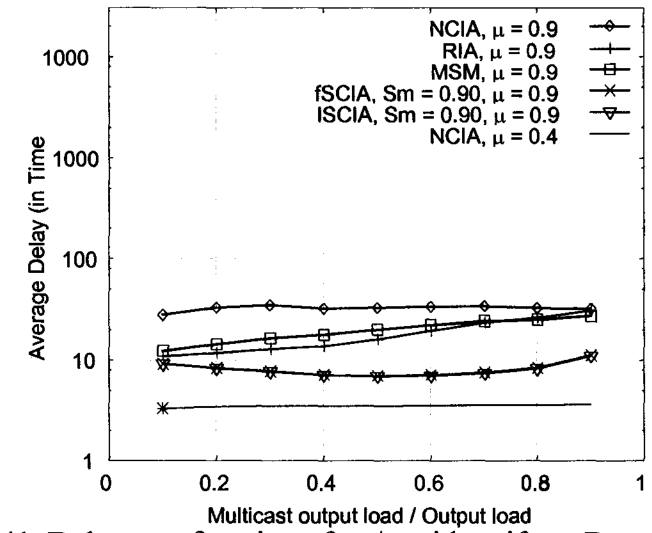


Figure 41. Delay as a function of μ_m/μ with uniform Bernoulli traffic, $F_{min} = 2$, $F_{max} = 6$, and $K = 4$.

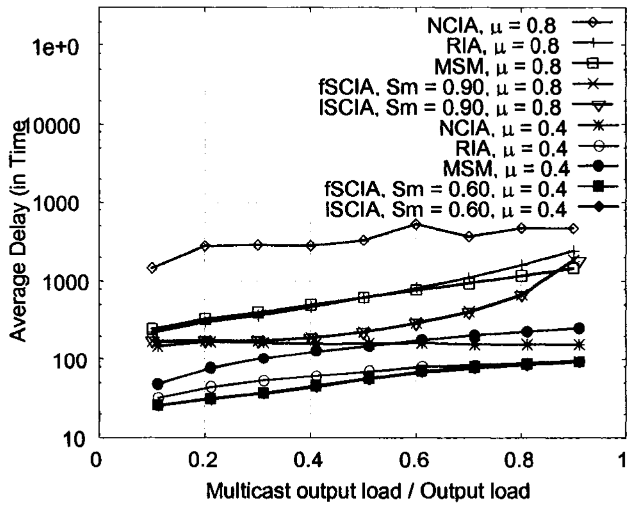


Figure 42. Delay as a function of μ_m/μ with uniform Bursty traffic, $E[b] = 32$, $F_{min} = 2$, $F_{max} = 6$, and $K = 16$.

5.4.1.2 Non-uniform Traffic

The non-uniform traffic model adopted in simulations is as follows: one of the N inputs, for instance input i, has a higher link arrival rate than others, and the other inputs have equal link arrival rates. Let Ω represent the ratio of λ_i to λ_j, where $j \neq i$. Figures 43 and 44 plot the average delay and packet loss ratio as a function of output load for non-uniform bursty traffic with an Ω of two. Although *l*SCIA and *f*SCIA are mainly proposed for uniform traffic, compared with other integration algorithms, they still perform better, i.e., smaller average delay and less average packet loss ratio are achieved under a given traffic pattern.

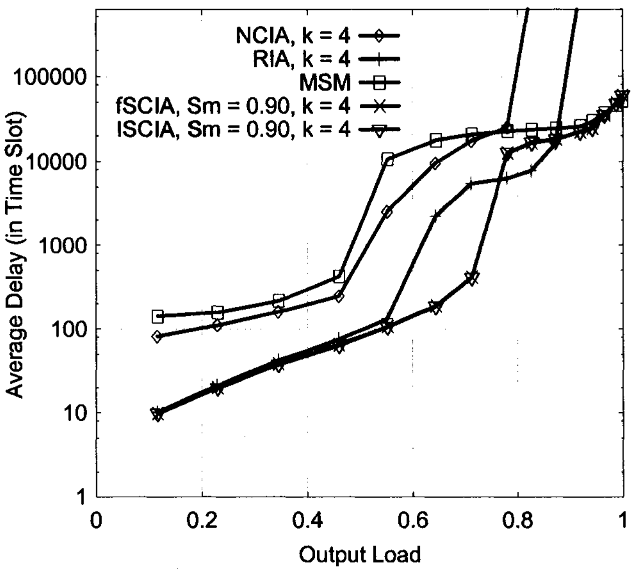


Figure 43. Delay as a function of normalized output load with non-uniform Bursty traffic, $\Omega = 2$, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, and $F_{max} = 6$.

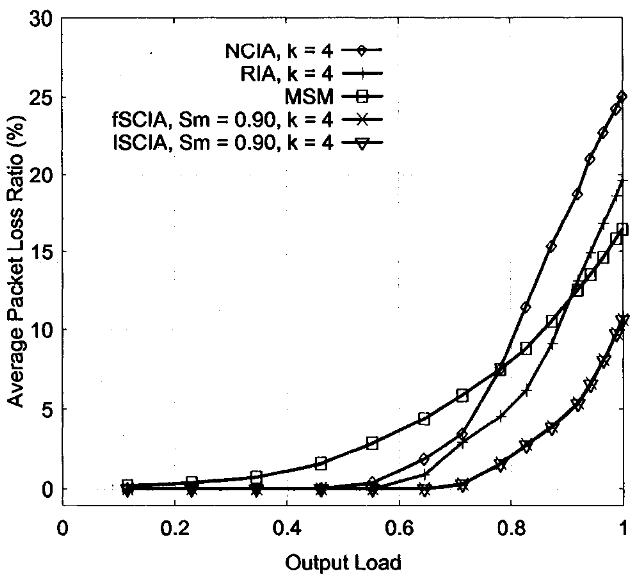


Figure 44. Packet loss ratio as a function of normalized output load with non-uniform Bursty traffic, $\Omega = 2$, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $L = 1000$.

5.4.2 Performance Improvement by Increasing K

Based on the aforementioned analysis, the working interval decreases while the saturation throughput of multicast scheduling decreases. Under some traffic patterns, the working interval may even be an empty set, if $\mu_{m\text{-}sat}$ is not big enough, which means that 100% throughput cannot be achieved under any S_m. Therefore, in order to enlarge the working interval, $\mu_{m\text{-}sat}$ needs to be increased. The saturation throughput of multicast scheduling can be increased efficiently through increasing the number of multicast queues (K). Figures 45 and 46 illustrate the improvement of delay performance introduced by increasing the number of multicast queues for Bernoulli traffic and bursty traffic, respectively. For the given traffic composition, S_m is fixed at a value close to μ_m/μ. The average delays as a function of output load are plotted. At first, the average delay performance improves dramatically as K increases. Once K is big enough such that 100% throughput is achieved, the improvement of delay performance is not significant.

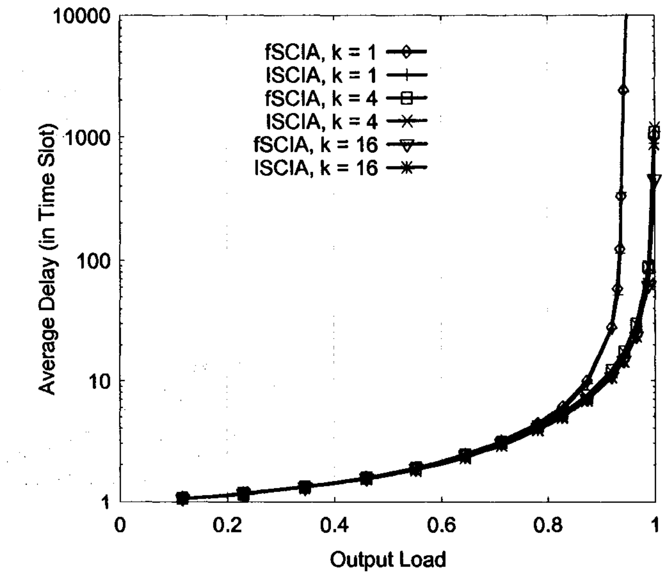


Figure 45. Delay as a function of normalized output load with uniform Bernoulli traffic, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $S_m = 0.73$.

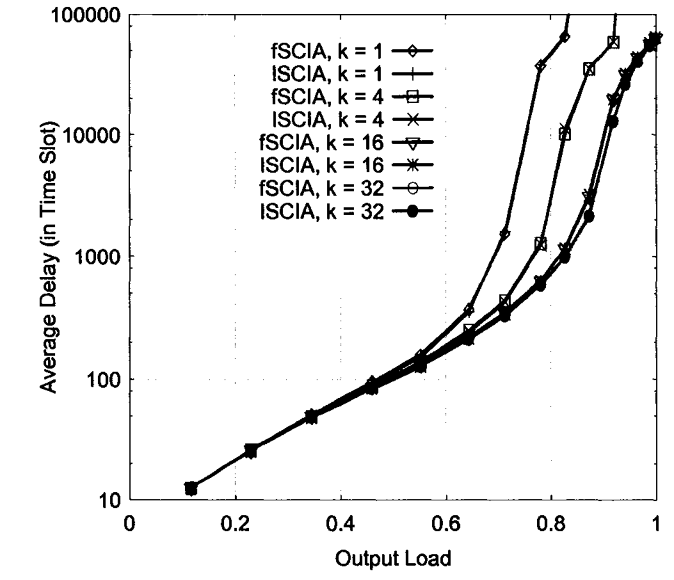


Figure 46. Delay as a function of normalized output load with uniform Bursty traffic, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $S_m = 0.75$.

The increase of the number of multicast queues leads to the decrease of the packet loss ratio, which is shown in Figures 47 and 48. The input buffer size is fixed at 100 packets for Bernoulli traffic and 1000 packets for bursty traffic, respectively. Like the delay performance, the performance of packet loss ratio improves significantly as the increase of K at first. Once K is big enough to achieve 100% throughput, the decrease of the packet loss ratio is almost negligible.

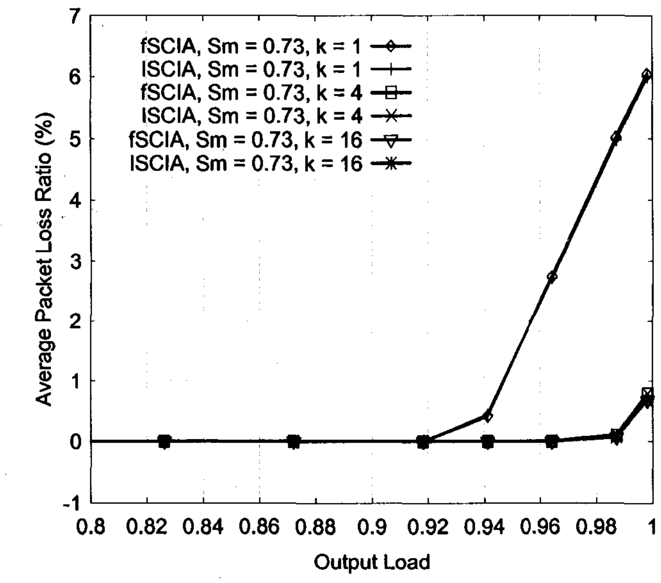


Figure 47. Packet loss ratio as a function of normalized output load with uniform Bernoulli traffic, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $L = 100$.

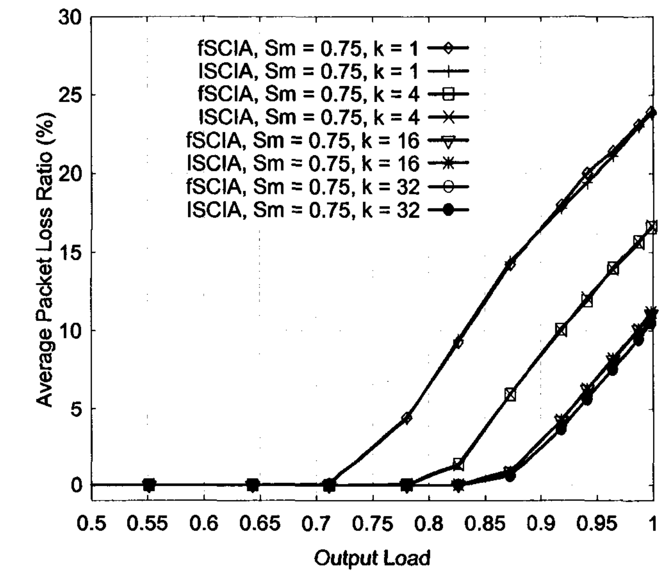


Figure 48. Packet loss ratio as a function of normalized output load with uniform Bursty traffic, $E[b] = 32$, $P_u = 70\%$, $\mu_m/\mu = 69.5\%$, $F_{min} = 2$, $F_{max} = 6$, and $L = 1000$.

Furthermore, the saturation throughputs as a function of K under different traffic compositions are shown in Figure 49. For a given μ_m/μ, S_m is fixed at a value close to μ_m/μ. The difference between S_m and μ_m/μ is less than 10% of μ_m/μ. With the same traffic pattern and the same S_m, saturation throughput increases and converges to 1 as K increases. If K is big enough, 100% throughput can be achieved.

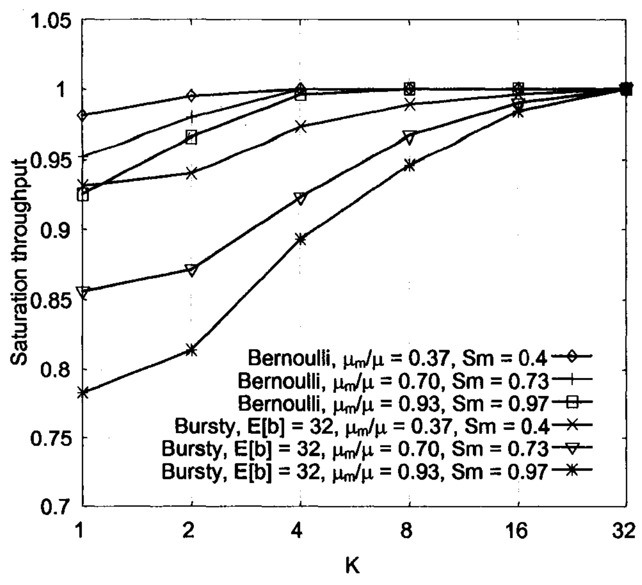


Figure 49. Saturation throughput as a function of K with uniform traffic, $F_{min} = 2$, $F_{max} = 6$ for *l*SCIA and *f*SCIA.

5.5 Summary

This chapter analyzed the design challenges to efficiently support both unicast and multicast traffic within one switching fabric. Simply considering multicast as a special case of unicast would not allow queueing and scheduling to be scalable. Likewise, simply considering unicast as a special case of multicast would compromise the performance of unicast traffic. Two efficient integration algorithms were proposed for input-queued switches with FIFO queues. Based on the theoretical analysis to the working interval of multicast service ratio, 100% throughput and a promising performance in terms of both delay and packet loss ratio can be achieved under variant traffic compositions. The analytical and simulation results show that the proposed integrated queueing and scheduling performs well in the present of diverse traffic patterns.

Since both unicast scheduling and multicast scheduling are based on algorithms designed specifically for uniform traffic, the two integration algorithms are proposed mainly for uniform traffic. However, the idea of providing separate queues for unicast and multicast traffic and coupling unicast and multicast scheduling in each time slot can be extended to design integration algorithms specifically for non-uniform traffic, which will be the future work.

CHAPTER VI

CONCLUDING REMARKS AND FUTURE RESEARCH

This chapter summarizes main contributions and conclusions in Section 6.1 and presents possible future research directions in Section 6.2.

6.1 Concluding Remarks

The contributions of this dissertation are listed below.

- Deduced the closed-form expressions of saturation throughput, average service time, and average delay for a large $N \times N$ multicast switch with multiple FIFO queues per input port;
- Created a Markov Chain model to analyze the probability distribution function of residue size at the beginning of a time slot and generalized the theoretical analysis on saturation throughput, average service time, and average delay to the case of $M \times N$ switches;
- Validated the theoretical analysis using a number of experimental data;
- Proposed two novel integrated scheduling algorithms, *l*SCIA and *f*SCIA, to integrate unicast and multicast scheduling within one switching fabric;
- Experimentally studied the performance of *l*SCIA and *f*SCIA and analyzed their properties.

The following conclusions are obtained through the research work presented in this dissertation.

- For a large $N \times N$ multicast switch with multiple FIFO queues per input port, a small number of queues (less than ten) are sufficient to achieve the sub-optimal performance;
- For a large $M \times N$ multicast switch with multiple FIFO queues per input port, a small number of queues (much less than $2^N - 1$) is a reasonable choice for the tradeoff between the saturation throughput and delay performances and the

scheduling overhead;

- For a large $M \times N$ multicast switch, the final achievable saturation throughput decreases as the ratio of M/N decreases;
- The performance analysis results for $N \times N$ or $M \times N$ multicast switches are valid for practical large-sized switches;
- The proposed integration algorithm, *l*SCIA and *f*SCIA, can achieve 100% throughput with theoretical guarantee and exhibit promising delay and packet loss ratio performances under both uniform Bernoulli and uniform bursty traffic.

6.2 Future Research Directions

There are several ways to extend this research, which are briefly discussed below.

6.2.1 Extending Performance Analysis of Multicast Switches to More General Cases

In order to facilitate the theoretical analysis on $N \times N$ or $M \times N$ multicast switches with K FIFO queues per input port given in Chapters III and VI, several assumptions are set up, some of which can be modified to extend the analysis to more general cases.

- *Small-sized and medium-sized switches*: In the analysis given in Chapters III and VI, the switch size (N) is assumed to be a very large number. The closed-form expressions for saturation throughput and delay are not relevant to N. Through simulations, it is shown that the analysis results is valid when $N > 80$. However, the experimental data also indicates that the saturation throughput drops obviously as N increases when N is not big enough (See Figures 4, 5, 20, and 21). By considering N as a factor during the analysis, both saturation throughput and delay will be functions of N. As a result, the analysis results will be valid for any-sized switches instead of just large-sized switches.
- *Finite queue size*: In Chapters III and VI, the queue size is assumed to be infinite and all the queues are modeled as $M/G/1$ queues. However, in the real-world switches and routers, the queue size is finite. Assuming that the queue size is K, the queues should be modeled as M/G/K queues. Then, in addition to saturation throughput and average delay, the close form expression for packet loss ratio also needs to be derived.

- *Non-random queueing policies and scheduling algorithms*: In Chapters III and VI, it is shown that the saturation throughput and delay performance cannot be noticeably improved by adding more queues when K is big enough. 100% throughput cannot be achieved especially when the value of M/N is small. Considering that random queueing policy and random scheduling algorithm are adopted, the performance can be further improved by choosing proper queue policy and scheduling algorithm. As what is discussed in Chapter II, queueing policies such as Majority [12] and LBQ [4] and scheduling algorithms such as MaxService [12] and GMSS [4] have been proposed to achieve promising performance for multicast traffic. The experimental results specified in Chapter V also suggest that WBA-MQ performs well for both Bernoulli and bursty traffic (See Tables 11 and 12). Analyzing multicast switches that deploy these policies and algorithms promises to be an exciting topic for future research.
- *Non-uniform traffic*: In Chapters III and VI, the multicast traffic is assumed to uniformly distribute among input and output ports. Nevertheless, in the real world applications, the traffic tends to be non-uniform. Thus analyzing system performances under non-uniform multicast traffic is very meaningful for the design of switches and routers that construct today's Internet.

6.2.2 Designing Multicast Scheduling Algorithms for Non-Uniform Traffic

For unicast traffic, LQF [34], OCF [34], and LPF [36] are designed to achieve 100% throughput for both uniform and non-uniform traffic. On the contrary, current multicast queueing policies and scheduling algorithms are mainly proposed for uniform traffic. And their performance is evaluated only under the assumption of uniform traffic. The queueing and scheduling of non-uniform multicast traffic remains an open issue. As what is mentioned previously, non-uniform multicast traffic is typical in the real world applications. Therefore, the design of queueing policy and scheduling algorithm for non-uniform traffic is an attractive direction for future research work. Here are some specific suggestions.

- Generally speaking, while pursuing high performance in terms of saturation throughput, delay, and packet loss ratio, this design also needs to assure stability, fairness, and starvation free.

- The multicast flows with different arrival rates should be evenly partitioned into input queues such that the output load of an input queue equals to each other.
- A certain number of flows with heaviest arrival rates should be split into different queues in order to mitigate HOL blocking.
- At an input port, the packets destined to the output ports with heavy output loads should be scheduled with higher priority. Meanwhile the packets destined to the output ports with light output loads cannot be starved.
- At an output port, the packets coming from the input ports with heavy input loads should be scheduled with higher priority. Meanwhile the packets coming from the input ports with light input loads cannot be starved.
- Scheduling decisions can utilize several system parameters as the reference, such as the queue length, the age of the HOL packet, the residue size of a HOL packet, and so on.

6.2.3 Theoretically Analyzing the Performance of Integrated Unicast and Multicast Scheduling

The performance of the integration algorithms proposed in Chapter V is mainly evaluated throughput simulations. Although simulation is a very important and useful tool in the area of queuing and scheduling of switches, it will be more convincible if theoretical support can be provided. Furthermore, the closed-formed expressions of performance metrics can describe the performance of the integration algorithms more comprehensively than sampled experimental data. Thus, the next direction for this research will be deriving the probability characteristics of service time, saturation throughput, delay, and packet loss ratio of the integrated algorithms. The analysis is expected to utilize the $M/G/1$ or $M/G/K$ model and the model of Markov Chain.

REFERENCES

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, November 1993, pp. 319–352.

[2] M. Andrews, S. Khanna, and K. Kumaran, "Integrated Scheduling of Unicast and Multicast Traffic in an Input-Queued Switch," *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'99)*, March 1999, pp. 1144–1151.

[3] S. A. Bassiouny, M. A. Abou-Of, W. A. El-Haweet, and M. N. El-Derini, "Fast-Multicast Parallel-Banyan Based ATM Switch," *Proceedings of 2003 International Conference on Communication Technology (ICCT 2003)*, vol. 1, April 2003, pp. 525–531.

[4] A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and C. Piglione, "On the Number of Input Queues to Efficiently Support Multicast Traffic in Input Queued Switches," *Proceedings of High-Performance Switching and Routing 2003 (HPSR 2003)*, Torino, Italy, June 2003, pp. 111–116.

[5] X. Chen, J. F. Hayes, and M. K. Mehmet-Ali, "Performance Comparison of Two Input Access Methods for a Multicast Switch," *IEEE Transactions on Communications*, vol. 42, no. 5, May 1994, pp. 2174–2177.

[6] W.-T. Chen, C.-F. Huang, Y.-L. Chang, and W.-Y. Hwang, "An Efficient Packet-Scheduling Algorithm for Multicast ATM Switching Systems," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, August 2000, pp. 517–525.

[7] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, "Measurement and Analysis of a Streaming-Media Workload," *Proceedings of 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01)*, San Francisco, California, March 2001, pp. 1–12.

[8] F. M. Chiussi and A. Francini, "Scalable Electronic Packet Switches," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, May 2003, pp. 486–500.

[9] F. M. Chiussi, Y. Xia, and V. P. Kumar, "Performance of Shared-Memory Switches Under Multicast Bursty Traffic," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, April 1997, pp. 473–487.

[10] J. Chung, M. Claypool, and Y. Zhu "Measurement of the Congestion Responsiveness of Realplayer Streaming Video over UDP," *Proceedings of the 13th International Packet Video Workshop (PV 2003)*, Nantes, France, April 2003.

[11] B. Dutson, C. Dutson, and S. Drayson, "New opportunities in streaming media report," *In Vision Consultancy Group*, September 2000.

[12] S. Gupta and A. Aziz, "Multicast Scheduling for Switches with Multiple Input-queues," *Proceedings of the 10th Symposium on High Performance Interconnects Hot Interconnects*, Stanford, CA, August 2002, pp. 28–33.

[13] J. F. Hayes, R. Breault, and M. K. Mehmet-Ali, "Performance Analysis of a Multicast Switch," *IEEE Transactions on Communications*, vol. 39, no. 4, April 1991, pp. 581–587.

[14] J. Y. Hui and T. Renner, "Queueing Analysis for Multicast Packet Switching," *IEEE Transactions on Communications*, vol. 42, February/March/April 1994, pp.723–731.

[15] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, Boston, 1990.

[16] S. Iyer and N. McKeown, "On the Speedup Required for a Multicast Parallel Packet Switch," *IEEE Communications Letters*, vol. 5, no. 6, June 2001, pp. 269–271.

[17] W. Kabacinski and G. Danilewicz, "Wide-Sense and Strict-Sense Nonblocking Operation of Multicast Multi-$\log_2^n$ Switching Networks," *IEEE Transactions on Communications*, vol. 50, no. 6, June 2002, pp. 1025–1036.

[18] R. Kannan and S. Ray, "MSXmin: A Modular Multicast ATM Packet Switch with Low Delay and Hardware Complexity," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, June 2000, pp. 407–418.

[19] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, vol. 35, no. 12, December 1987, pp. 1347–1356.

[20] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE*

Communications Magazine, vol. 36, no. 5, May 1998, pp. 144–151.

[21] M.S.A. Khan, S.M.A. Burney, and M. Naseem, "Multislot Scheduling Algorithm in ATM Networks," *Proceedings of 2004 International Conference on Networking and Communication (INCC 2004)*, June 2004, pp. 89–94.

[22] H. Kim and K. Kim, "Performance Analysis of the Multiple Input-Queued Packet Switch with the Restricted Rule," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, June 2003, pp. 478–487.

[23] T. Kuang and C. Williamson, "Hierarchical Analysis of RealMedia Streaming Traffic on an IEEE 802.11b Wireless LAN," *Computer Communications*, Elsevier, vol. 27, no. 6, June 2004, pp. 538–548.

[24] K. L. E. Law, A. Leon-Garcia, "A large scalable ATM multicast switch," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, June 1997, pp. 844–854.

[25] S. Lee, D. Cho, "Packet-scheduling Algorithm Based on Priority of Separate Buffers for Unicast and Multicast Services," *Electronic Letters*, vol. 39, no. 2, 23rd January 2003, pp. 259–260.

[26] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on Average Delays and Queue Size Averages and Variances in Input Queued Cell-Based Switches," *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2001)*, Anchorage, Alaska, April 2001, pp. 1095–1103.

[27] C. Li, H. M. Heys, and R. Venkatesan, "Design and Implementation of the Scalable Multicast Balanced Gamma (BG) Switch," *Proceedings of Eleventh International Conference on Computer Communications and Networks (ICCCN 2002)*, October 2002, pp. 518–521.

[28] S. C. Liew, "A General Packet Replication Scheme for Multicasting with Application to Shuffle-Exchange Networks," *IEEE Transactions on Communications*, vol. 44, no. 8, August 1996, pp. 1021–1033.

[29] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "On the Throughput of Input-Queued Packet-Based Switches with Multicast Traffic," *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and*

Communications Societies (IEEE INFOCOM 2001), Alaska, April 2001, pp. 1664–1672 .

[30] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast Traffic in Input-Queued Switches: Optimal Scheduling and Maximum Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, June 2003, pp 465–477.

[31] N. McKeown, "A Fast Switched Backplane for a Gigabit Switched Router," *Business Communications Review*, vol. 27, no. 12, 1997.

[32] N. McKeown, "The *i*SLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions on Networking*, vol.7, no.2, April 1999, pp. 188–201.

[33] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellesick, and M. Horowitz, "The Tiny Tera: A Packet Switch Core," *IEEE Micro*, vol. 17, no. 1, February 1997, pp. 26–33.

[34] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, vol. 47, no. 8, August 1999, pp. 1260–1272.

[35] M. K. Mehmet-Ali, and S. Yang, "Performance Analysis of a Random Packet Selection Policy for Multicast Switching," *IEEE Transactions on Communications*, vol. 44, no. 3, March 1996, pp. 388–398.

[36] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *Proceedings of Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'98)*, vol. 2, 29 March–April, 1998, pp. 792–799.

[37] J. V. der Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact," *Proceedings of 7th International Workshop on Web Content Caching and Distribution*, Boulder, Colorado, August 2002.

[38] C. Minkenberg, "Integrating Unicast and Multicast Traffic Scheduling in a Combined Input- and Output-Queued Packet-Switching System," *Proceedings of Ninth International Conference on Computer Communications and Networks (ICCCN 2000)*, Las Vegas, NV, October 2000, pp. 127–134.

[39] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of Nonblocking ATM Switches with Multiple Input Queues," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1,

February 1999, pp. 60–74.

[40] D. Pan and Y. Yang, "FIFO-Based Multicast Scheduling Algorithm for Virtual Output Queued Packet Switches," *IEEE Transactions on Computers*, vol. 54, no. 10, October 2005, pp. 1283–1297.

[41] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes,* 4th ed., McGraw-Hill, New York, 2002.

[42] C. Partridge *et al*, "A 50-Gb/s IP router," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, June 1998, pp. 237–248.

[43] D. Plonka, "UW-Madison Napster Traffic Measurement," URL: http://net.doit.wisc.edu/data/Napster, March 2000

[44] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast Scheduling for Input-Queued Switches", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, June 1997, pp. 855–866.

[45] B. Sikdar and D. Manjunath, "Queueing Analysis of Scheduling Policies in Copy Networks of Space-Based Multicast Packet Switches," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, June 2000, pp. 396–406.

[46] M. Song and M. Alam, "Two Scheduling Algorithms for Input-queued Switches Guaranteeing Voice QoS," *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2001)*, Texas, vol. 1, November 2001, pp. 92–96.

[47] M. Song, S. Shetty, G. Loaisiga, and H. J. Yang, "Efficient Queueing Scheme for Multicast Switches in Overlay Networks," *Proceedings of the 16th International Conference on Parallel and Distributed Computing Systems '03*, Reno, Nevada, August 2003, pp. 76–81.

[48] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Sicily, Italy, October 2004, pp. 41–54.

[49] D. Stiliadis, "Efficient Multicast Algorithms for High-Speed Routers", *Proceedings of IEEE workshop on High Performance Switching and Routing (HPSR 2003)*, Torino, Italy, June 2003, pp. 117–122.

[50] Y. Tamir and G. L. Frazier, "High-Performance Multiqueue Buffers for VLSI Communication Switches," *Proceedings of 15th Annual International Symposium*

on Computer Architecture, May-June 1988, pp. 343–354.

[51] R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Pennsylvania, November 1983.

[52] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy, "Organization-Based Analysis of Web-Object Sharing and Caching," *Proceedings of 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99)*, Boulder, Colorado, October 1999.

[53] Y. Yang and J. Wang, "A New Design for Wide-Sense Nonblocking Multicast Switching Networks," *IEEE Transactions on Communications*, vol. 53, no. 3, March 2005, pp. 497–504.

[54] SIM++ simulator, URL: http://www.odu.edu/engr/networking/tools.html.

VITA

Weiying Zhu
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529

EDUCATION

Master of Engineering in Communication and Information System, Huazhong University of Science and Technology, Wuhan, Hubei, P. R. China, June 1999

Bachelor of Engineering in Biomedical Electrical Engineering, Xi'an Jiaotong University, Xi'an, Shanxi, P. R. China, July 1996

PROFESSIONAL CHRONOLOGY

Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, Virginia

Graduate Research Assistant, January 2003 to present

Graduate Teaching Assistant, January 2003 to August 2005

Bell Labs China, Lucent Technologies Co. Ltd, Beijing, P. R. China

Member of Technical Staff (Software Engineer), July 1999 to January 2003

Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, P. R. China

Graduate Research Assistant, September 1996 to June 1999

HONORS AND AWARDS

- The First Place in Graduate Student Poster Competition at Research Expo 2006, sponsored by Eastern Virginia Medical School, Norfolk State University and Old Dominion University, April 2006
- ODU Dissertation Fellowship Award, Old Dominion University, Academic Year 2005–2006

- ECE Ph.D. Research Assistant Award, Department of Electrical and Computer Engineering, Old Dominion University, 2004
- Member of Bell Labs President's Gold Winner Team, Lucent Technologies, 2000
- Outstanding Thesis Award, Huazhong University of Science and Technology, 1999
- Outstanding Alumna Award, Huazhong University of Science and Technology, 1999
- Outstanding Graduate Student Awards, Huazhong University of Science and Technology, 1998 and 1997
- Outstanding Undergraduate Student Awards, Xi'an Jiaotong University , 1995 and 1993
- Sanhao Model Award, Xi'an Jiaotong University, 1994
- *General Physics* Contest Award, Xi'an Jiaotong University, 1994
- *Advanced Mathematics* Contest Award, Xi'an Jiaotong University, 1993

PUBLICATIONS

Papers submitted for review

[1] Weiying Zhu and Min Song, "Performance Analysis of Large Multicast Packet Switches with Multiple Input Queues and Gathered Traffic," *IEEE/ACM Transactions on Networking*, submitted for review, August 2005.

Journal publications

[2] Weiying Zhu and Min Song, "Integration of Unicast and Multicast Scheduling in Input-Queued Packet Switches," *Computer Networks*, Elsevier, accepted June 2005, to appear.

[3] Min Song, Weiying Zhu, Andrea Fancini, and Mansoor Alam, "Performance Analysis of Large Multicast Switches with Multicast Virtual Output Queues," *Computer Communications*, Elsevier, Volume 28, Issue 2, Feb. 2005, pp. 189–198.

[4] Min Song and Weiying Zhu, "Throughput Analysis for a Multicast Switch with Two Input Queues," *IEEE Communications Letters*, July 2004, pp. 479–481.

[5] Weiying Zhu, Bingxin Shi, and Ling Zhou, "The application of discrete event simulation method based on object-oriented programming on the simulation of LAN," *Mini-Micro Systems*, Shenyang, China, June 1999, pp. 433–437.

[6] Weiying Zhu and Bingxin Shi, "TCP/IP communication using WinInet," *Computers & Communications*, Beijing, China, November 1998, pp. 70–74.

Conference publications

[7] Weiying Zhu, Min Song, and Stephan Olariu, "Quality of Service Routing with Stability Estimation in Mobile Ad-Hoc Networks," accepted by *Fourteenth IEEE International Workshop on Quality of Service (IWQoS 2006)*, Yale University, New Haven, CT, June 2006.

[8] Min Song and Weiying Zhu, "Integrated Queuing and Scheduling for Unicast and Multicast Traffic in Input-Queued Packet Switches," in *Proceedings of the 2nd IASTED International Conference on Communication and Computer Networks (CCN 2004)*, MIT, Cambridge, MA, November 8–10, 2004.

[9] Min Song and Weiying Zhu, "Delay Analysis of Multicast Switches with Multiple Input Queues," in *Proceedings of the 2004 IEEE Workshop on High Performance Switching and Routing (HPSR 2004)*, Phoenix, AZ, April 18–21, 2004.

[10] Min Song, Sachin Shetty, and Weiying Zhu, "Evolutionary Programming in a Distributed Scheduler Architecture," in *Proceedings of 16th International Conference on Computer Applications in Industry and Engineering*, Las Vegas, NV, November, 2003.