ΩSWITCH

A HIGH SPEED ATM SWITCH FOR MULTICAST AND UNICAST

APPROVED BY SUPERVISING COMMITTEE:

Dr. Rajendra V. Boppana, Chair

Dr. Robert E. Hiromoto

Dr. Weining Zhang

Accepted:

Dean of Graduate Studies

ΩSWITCH

A HIGH SPEED ATM SWITCH FOR MULTICAST AND UNICAST

by

RAMAKANTH GUNUGANTI, B.Tech.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences and Engineering
Division of Computer Science
August 2000

Acknowledgements

The thesis work has been partially supported by DOD/AFOSR grant F49620-96-1-0472 and by support from the University of Texas at San Antonio Division of Computer Science.

August 2000

ΩSWITCH
A HIGH SPEED ATM SWITCH FOR MULTICAST AND UNICAST

Ramakanth Gunuganti, B.Tech.
The University of Texas at San Antonio, 2000

Supervising Professor: Rajendra V. Boppana

Cell-based ATM switches are critical components of future internets providing diverse services and rigorous QoS guarantees for multimedia multicast and unicast connections. These switches can use input buffering, output buffering or a combination of both. While input-buffered switches are attractive for unicast connections, output-buffered switches offer the flexibility needed to handle multicasts. In this thesis, we investigate a combination of input and output buffering to reduce the switch complexity and increase its scalability.

We propose a new switch, called the Ωswitch, which uses a combination of input and output buffering to provide superior performance for unicast and multicast communication. The Ωswitch is based on two copies of banyan networks or one banyan network operating at twice the speed of the line rates. Furthermore, the switch fabric does not use internal buffering, in contrast to the current solutions which use crossbars or banyans with internal buffering. Cell selection mechanism is simple and can be implemented in hardware. Since a speedup of only 2 is required and banyan networks are less expensive than crossbars, this design is scalable in terms of speed and cost. Our simulation results demonstrate that the Ωswitch gives superior performance for pure multicast traffic and a comparable performance for pure unicast traffic compared to the current solutions. The Ωswitch also gives nearly 100% switch utilization with combined unicast and multicast traffic, while the current input-queued, crossbar-based switch designs give about $70 - 80\%$ utilization. Furthermore, the Ωswitch gives very high multicast performance even when no fan-out splitting is used.

Contents

List of Figures

Chapter 1

Introduction

The demand for network bandwidth has been increasing at an enormous rate with the exponential increase in the number of users on the Internet. The bandwidth problem can be alleviated by using high speed switched networks, consisting of high speed links and switches. With recent and continuing advances in technology, extremely high-speed optical links can be provided. However, switching the cells carried and delivered by optical links is a major challenge. Many of the services spawned by the high-speed internets, require the delivery of the same information to multiple destinations and this type of communication is called multicast communication. Video/audio conferencing, resource discovery in a distributed network, live games, pay-per-view movies provide examples of such applications. As these applications are increasingly used on the Internet, the resulting multicast traffic will consume a significant portion of the network bandwidth. Multicast traffic can be handled using unicast routing mechanisms, by making multiple copies of the message, one for each destination, but this results in significant wastage of processing power and network bandwidth. To handle multicast communication efficiently, however, significant hardware (switches/routers) and software (routing protocols) support has to be provided.

In this thesis,we focus on the design of high-speed switches. In the next section we describe the existing switch designs. Then we describe the Ωswitch; our main contribution.

1.1 Related Work

The earliest switch fabric designs were based on time division multiplexing. This allowed a single fabric to be shared by all the input and outputs. The inputs get access to the switch fabric using simple round robin scheme or in some other pre-determined manner. The Atmosphericring switch from the RACE research program [26] and the ATOM switch from NEC [27], used time division switch fabrics. These switches could not handle the traffic as the line rates started to increase at an enormous rate. Thus, there was a need to increase the bandwidth inside the switch.

As an alternative, space division multiplexing based switches were proposed. These fabrics, such as the crossbar, provide multiple paths between the input and output ports, thus increasing the bandwidth inside the switch. Al the cells arriving at the inputs of a switch may be sent to its outputs if no two cells require the same output. Otherwise only some of the cells may be switched and the others need to be buffered and sent at a later time. Initially most of the space division fabric switches used input buffering ([2], [10], [11]). This allowed queuing of cells, that cannot be transmitted due to output conflicts, at the input ports. But, if only the cells at the head of the queue are served in a crossbar based switch, it was proved ([8]) that a maximum throughput of 58.6% can be achieved. The cells at the head of the queue block the other cells in the queue from being transmitted and this problem is called the Head Of Line (HOL) blocking problem. To overcome this, output queued switches ([3], [28], [32]) were proposed.

In an ideal output-queued switch, every output queue must be able to accept a cell from every input simultaneously. Thus an output-queued switch can be much more complex than an input-queued switch, as the fabric and the output queues must operate at a much higher rate than the line rate. The tandem fast banyan packet switch [28], knockout switch [3] and christmas tree switch [32] are some of the designs that use output queuing. But these designs become impractical and expensive to implement as the line rates and the number of input and output ports increase. Thus, some researchers have further investigated multistage network based switches and input-queued, crossbar-based switches. The multistage network based switches are cost-effective, but they require elaborate cell selection or cell routing schemes ([31]) or switch fabric designs ([25]) (with internal buffering) to transmit the cells. Though hardware based

schemes [1] have been proposed for efficient cell selection and routing for unicast, the current multicast scheduling requires complex designs [5].

Efficient scheduling policies ([37], [36], [11], [23]) and the use of virtual output queuing [35] are proposed to increase the throughput of input-queued, crossbar-based switches beyond the 58% limit imposed by HOL blocking. Scheduling policies like, Parallel Iterative matching [11], iSlip [23] for unicast scheduling and a simple round robin scheduling without fan-out splitting (RSIMPLE)[7], a weight based aging scheme called WBA [6], for multicast scheduling, was proposed to increase the performance of input-queued, crossbar-based switches. But, these scheduling algorithms increase the complexity of the otherwise simple input-queued switches. The multicast scheduling policies further complicate the switch designs, as they use a technique called fan-out splitting, that requires additional hardware at the inputs to make multiple copies of a multicast cell to be transmitted over multiple slot times. This further increases the complexity of input cell selection and scheduling hardware.

1.2 Contribution

Recently, there has been an increasing interest in combined input-output-queued switch designs [4] because of their simplicity and high performance. To take advantage of both input queuing and output queuing, we have used a combination of both these techniques to propose a combined input-output-queued switch design ([33], [34]), called the Ωswitch. The switch fabric consists of a banyan network operating at twice the line speed or two banyans running at the line speed. The switch fabric need not operate at much higher rates than the line rates as input queues can be used to hold back the cells with output conflicts. Hence the design is not only cost-effective but also is simple to implement.

The Ωswitch uses simple and efficient techniques to handle both unicast and multicast traffic (especially for multicast, without fan-out splitting). We present two simple multicast scheduling policies: random scheduling that uses fan-out splitting like WBA and round robin scheduling that does not use fan-out splitting like RSIMPLE. Using simulations we demonstrate that even the simplistic round robin policy on the Ωswitch performs better than WBA. It also outperforms pure input-queued designs with a

combination of unicast and multicast traffic by achieving close to 100% switch utilization at high loads. Furthermore it is easy to implement in hardware as it uses no multicast cell splitting.

1.3 Organization

The thesis is organized as follows. Chapter 2 presents a general cell-based switch design and describes in detail the unicast and multicast scheduling policies for input-queued, crossbar-based switches. Chapter 3 presents the Ωswitch design and describes unicast scheduling on it. Chapter 4 presents different multicast scheduling policies on the Ωswitch. Chapter 5 describes the simulation results and chapter 6 presents conclusions and future extensions.

Chapter 2

Cell Switch Designs

In this chapter, we describe switch designs for unicast and multicast communication. First, we describe the switching terminology used in this work.

Non-Blocking Switch A switch is said to be non-blocking if the switch fabric is internally non-blocking. If the cells presented at the inputs have output ports with no contentions, then they are all guaranteed to be delivered in such a fabric. e.g. Crossbars.

Blocking Switch A switch is said to be blocking if the switching fabric is internally blocking. If the cells presented at the inputs have output ports with no contentions, still it cannot be guaranteed that the cells will be delivered to the outputs.

Unicast Cell A fixed sized packet with both the input port and the output port specified. The cell has to be delivered to the output port specified.

Multicast Cell A fixed sized packet, similar to a unicast cell, except that the destination is a vector of output ports. The cell has to be delivered to all the outputs specified by the destination vector.

Speed Up It is the ratio of the speed at which the switch fabric operates and the input line rate.

Fan-out Fan-out of a multicast cell is the number of destinations an input cell has to be delivered to. It is the length of the destination vector.

No Fan-out Splitting All the copies of a cell, specified by the destination vector have to be sent in the same time slot. If a cell cannot be sent to one of the destinations specified in the destination vector due to contention, it is not delivered to any of the other destinations.

Fan-out Splitting A multicast cell can be delivered to the outputs specified by the destination vector in any number of cell times.

Slot Time The time taken by the switch fabric to transmit the cells presented at the inputs to the outputs.

2.1 High Speed switches

This section describes the main components of a high-speed switch. As shown in Figure 2.1, a high speed switch has the following components:

1. Input Queues
2. Output Queues
3. Switch fabric
4. Unicast Scheduler
5. Cell-splitter
6. Multicast Scheduler

Depending on the switch design, a switch can have only input queues or only output queues or both. Additional buffers can be used in the switch fabric, but we are not interested in such designs. Multicast scheduler and cell-splitter are used only if the switch supports multicasting.

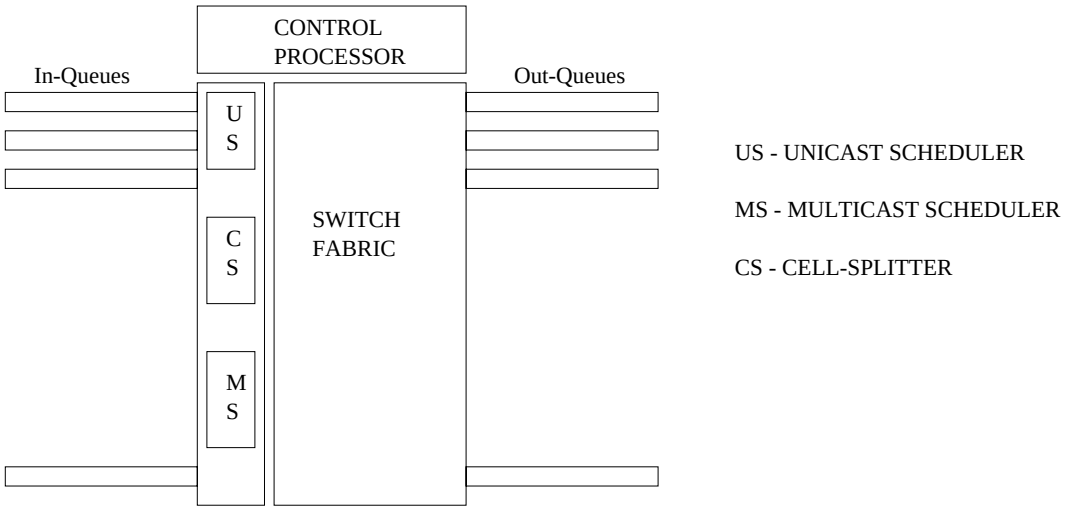

Figure 2.1: A High Speed Switch.

2.1.1 Input-Queued Switches

High Speed switches that queue the cells arriving at the input port are called input-queued switches. Later they are moved through the switch fabric by the scheduler. It has been proved in [8] that if FIFO queues are used at the inputs for crossbars then the maximum throughput that can achieved is 58% due to HOL (Head of Line) blocking. If a cell at the head of the queue cannot go through the switch due to contention, all the cells to different destinations in the queue are blocked. It has also been shown by [11, 9], [11] - [19] that, by using non FIFO queues and good scheduling policies, the utilization can be increased considerably.

2.1.2 Output-Queued switches

To overcome the HOL problems associated with input queuing, output queuing was proposed [3]. Multiple cells can be transmitted at the same time to the same output port and queued for transmission on the output link. In a typical high speed switch with M input ports and N output ports, the switch fabric has to operate at M times the line rate or use M copies of the same network to transmit the cells. The output queues also must operate at a much higher rate than the line rates. This approach does not

scale with the increase of input ports, since the memory bandwidth does not increase in proportion to the network bandwidth.

2.1.3 Switch Fabric

The switch fabric is the most important component of any high-speed switch. It effects the cost, scalability and complexity of the switch design. A switch fabric can use either time division multiplexing or space division multiplexing [20]. In the first case, time division multiplexing, all the input and output ports share a single communication channel to transfer cells. A fixed round-robin scheme is used to distribute the channel among various ports. In the second case, space division multiplexing, multiple paths are provided between the input and output ports. These paths can be used concurrently and hence multiple cells can be transmitted across the switch fabric at the same time. Most of the research work [8, 21, 2, 1], for the past few years has been on space division fabrics, as it provides more bandwidth.

Based on the switch fabric design([29], [30]), a switch can be considered as non-blocking or blocking. In non-blocking switches, all the cells presented to the switching fabric with non-conflicting outputs are guaranteed to be delivered at the output ports. In blocking networks cells destined to different outputs can contend for resources inside the switching fabric and hence no such guarantees can be made. A typical example of a non-blocking switch fabric is the crossbar and a blocking switch fabric is banyan network. A blocking switch fabric is more expensive than a non-blocking switch fabric as it requires more switching elements and the switching elements used are complex.

A crossbar switch employs a grid of switching elements as shown in Figure 2.2. It is a non-blocking network as a connection of an input to an output does not block the connection of any other input to any other output. The total number of switching elements required to implement a crossbar with M input ports and N output ports is $\Theta(MN)$. The complexity of the switching network grows as $\Omega(M^2)$, increasing exponentially with M. Hence the switching network is practically unrealizable as the number of input ports increase. Some of the switches based on crossbar designs are the Knockout switch [3] and Tiny Tera switch [2].

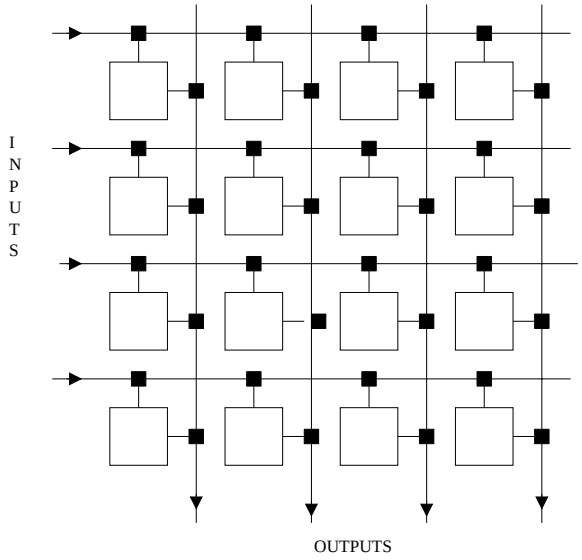

Figure 2.2: 4x4 Crossbar Network

A banyan network is used to describe a family of self-routing networks constructed from smaller switching elements with a single path between any input output pair. This network has $\log_2 M$ stages, where M is the number of inputs. Each stage of the network consists of an interconnection pattern that connects M inputs to M outputs. A link exists between input i and output j if the following is true:

$$\begin{aligned} j &= 2 * i, \qquad 0 \leq i \leq M/2 - 1, \\ & \quad 2 * i + 1 - M, \quad M/2 \leq i \leq M - 1. \end{aligned}$$

This interconnection pattern is also called a perfect shuffle [22]. Figure 2.3a shows such an interconnection pattern for 8 inputs and 8 outputs. In each stage of a banyan network, a perfect shuffle interconnection pattern feeds into a set of $M/2$ switching elements. Each switching element is in one of the two modes. In one mode (Figure 2.3b), the inputs are sent straight to the outputs called the pass through connection. In the other mode (Figure 2.3c), the inputs to the switching elements are crossed over and then sent out, called the cross-over connection. The banyan network has $M/2 \log_2 M$ switching elements and the cost of this network grows as $Mlog_2 M$. This is considerably less than the M^2 cost of the crossbar switch. Hence banyan networks are more scalable in terms of cost than the crossbars. The main drawback of the banyan networks is that they are blocking networks and hence require tedious

cell selection and cell routing techniques or buffers at the switching elements or else the performance drops significantly under high loads.

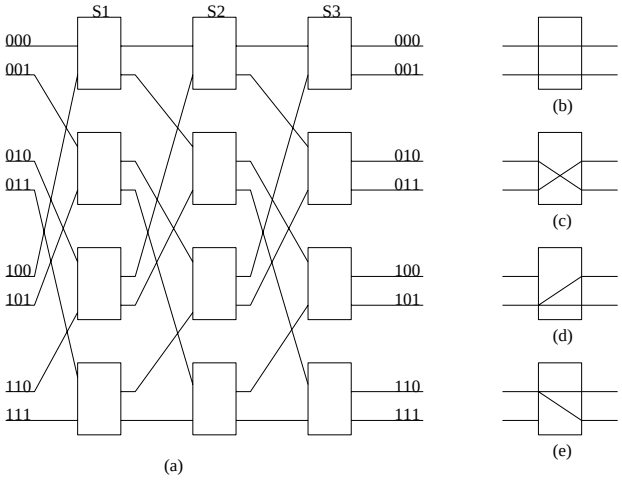

Figure 2.3: (a) 8x8 Omega Network, (b) Pass-through connection, (c) Cross-over connection, (d,e) Possible connections for multicasting.

These networks are also called self-routing networks. At each stage, the switching elements examine the bit in the output port that corresponds to that stage, starting with the most significant bit for the first stage. If that bit is 1, the cell is routed to the lower output or else the cell is routed to the upper output. Some of the switch designs based on banyan networks are the Phoenix switch [31], network hardware specific selection network based switch [1], Multinet switch [25] and Buffered MIN structure based switch [5].

2.1.4 Unicast Scheduler

The unicast scheduler is placed between the input queues and the switching fabric. The fabric operates under the scheduling algorithm implemented by the scheduler. It supplies the cells that have to be transferred, to the switching fabric by examining the input queues. Different scheduling policies can be implemented in the scheduler. Scheduler plays a very important role when input queuing is used. To overcome the effects of HOL blocking, the scheduler may have to examine multiple input queues or examine deep into the input queues. Many scheduling policies and switch fabrics have been proposed

in the literature for both crossbars and banyans. In this section we briefly present some of the unicast scheduling policies for crossbars and banyan networks and present some issues for multicast scheduling.

Unicast Scheduling Policies for Input-Queued Crossbars

Exhaustive lookup techniques (looking ahead into the FIFO queues [16], [18]) and Virtual Output Queuing [35] are two most popular techniques used to overcome the HOL blocking problem in input queued switches. The first technique involves looking deep into the input FIFO queue when the cell at the head of the queue cannot be sent during that cycle. Though it improves the performance, it is still sensitive to traffic arrivals and does not perform well for bursty arrivals. To eliminate the HOL blocking completely virtual output queuing was proposed by Tamir et al. in [35]. Rather than using a single FIFO queue for all the cells, each input maintains a separate FIFO queue for all the outputs. The importance of scheduling policies has grown further with these virtual output queues.

Many scheduling algorithms, maximum matching algorithms ([15], [10], [16]), were proposed to increase the throughput and reduce the delays by taking advantage of the virtual output queuing technique. Many of these algorithms are heuristic algorithms attempting to maximize the number of connections in each cell time and hence maximizing the instantaneous allocation of bandwidth. But, most of these algorithms, either starve one of the virtual output queues, or, are too complex to implement in hardware. Anderson et al. [11], proposed a Parallel Iterative Matching algorithm to achieve a throughput of 95% for crossbar-based switches. They introduced randomness into the algorithm to avoid starvation. The algorithm is based on a series of request, grant and accept phases. In the request phase, each unmatched input sends a request to every output for which it has a cell. In the grant phase, the outputs choose an input randomly for the requests. In the accept phase, the inputs choose a grant randomly if they receive multiple grants. Using randomness they were able to prevent starvation of some of the inputs. But this algorithm is difficult and expensive to implement in hardware. To overcome some of the drawbacks of randomness Nick Mckeown et al., proposed a round robin based matching algorithm called islip [23] for crossbar-based switches. This algorithm not only achieves close to 100% throughput but is also easier to

implement in hardware compared to Parallel Iterative Matching algorithm as the arbiters used for round robin scheme are much simpler than the arbiters used for random scheme.

All the algorithms proposed for crossbars are based on the non-blocking nature of the crossbar networks. These algorithms do not work for banyans as the banyan networks are blocking networks. To overcome the blocking nature of the banyan networks, the banyan network based switches require elaborate routing and cell selection mechanisms [13], [1]. Many software based schemes exist to set up the paths in these networks but these schemes are either too complex or do not scale well with increase in number of ports. Recently a hardware based selection technique was proposed by Boppana et al. [1], that is simple to implement and scales well with the increase in number of ports.

2.1.5 Cell-Splitter

A cell-splitter is used by most of the banyan and crossbar-based switches to implement multicast scheduling. Multicast scheduling is much more complex than unicast, as a single multicast cell has to be delivered to multiple destinations. Different strategies have been proposed to handle multicast traffic. The simplest among all is to make multiple copies of multicast cells and handle them like unicast, but this results in wasting resources and bandwidth. There are two other ways to handle multicast traffic namely, no fan-out splitting and fan-out splitting. In the first case, either all the copies of the cell are transmitted in a single slot time or none of them are transmitted. That is, if any of the output cells loses contention for an output port, none of them is transmitted. This scheme is simple to implement in hardware. In the second case, output cells may be delivered to output ports over any number of cell slot times. If during a slot time, the output cell can be delivered to some of the destinations specified in the destination vector of the multicast cell, then two copies of the multicast cell are made with new destination vectors. The first copy contains the destinations that cannot be reached during the slot time and the second copy has the destinations that can be reached successfully during the time slot. The first copy remains in the input queue while the second copy is sent to the switching fabric to be transmitted. A cell-splitter implements a multicast scheduling algorithm that uses fan-out splitting, and depending

on the requirement for copies of a multicast cell, splits the destination vector and makes multiple copies of the same cell with different destination vectors.

The use of cell-splitter adds additional complexity to a switch design. But it is required in most of the switch designs as pointed out by earlier researchers [6, 7]. They have shown that fan-out splitting is work conserving for input-queued, crossbar-based switches. The use of cell-splitting can be averted by providing additional bandwidth inside the switch fabric. This can be done by operating the switch fabric at higher speeds or by using multiple copies of the underlying network. We explore this approach further in the later chapters by using it in the design of a new switch based on banyan networks. A cell-splitter works in close collaboration with the multicast scheduler and in the next section we describe in detail the multicast scheduling policies used for crossbars.

2.1.6 Scheduling Multicast Cells for Crossbars

In this section we present multicast scheduling policies for input-queued crossbars. First, we present a couple of techniques previously proposed in the literature. Later on, we present some new techniques based on these techniques.

McKeown et al., [6] have proposed several multicast scheduling algorithms such as: TATRA; WBA; Concentrate; Distribute; and compared their performances. It was shown that WBA gives the best performance of all and is easier to implement in hardware. Sanjeev Khanna et. al. [7], have proposed a scheme to integrate unicast and multicast traffic in an input queued switch. They use round robin scheme for multicast scheduling. We present these two algorithms in the following subsections.

Weight Based Algorithm

Weight Based Algorithm (WBA)[6] algorithm works by assigning weights to input cells based on their age and fan-out. Age is the number of cell slot times an input cell has to wait at the input before being transmitted, and fan-out is the number of destination ports the cell is supposed to be delivered to. Inputs send their weights to output ports. Among all the inputs competing for a particular output,

the output port chooses the heaviest input port. Based on their previous experience, they found that to achieve fairness a positive weight has to be given to the age and a negative weight to the fan-out (the older the heavier and the larger the lighter). If $-f$ is the weight assigned to fan-out and a is the weight assigned to age, then no cell waits at the input port for more than $M + fN/a - 1$ cell times. Where M is the number of input ports and N is the number of output ports.

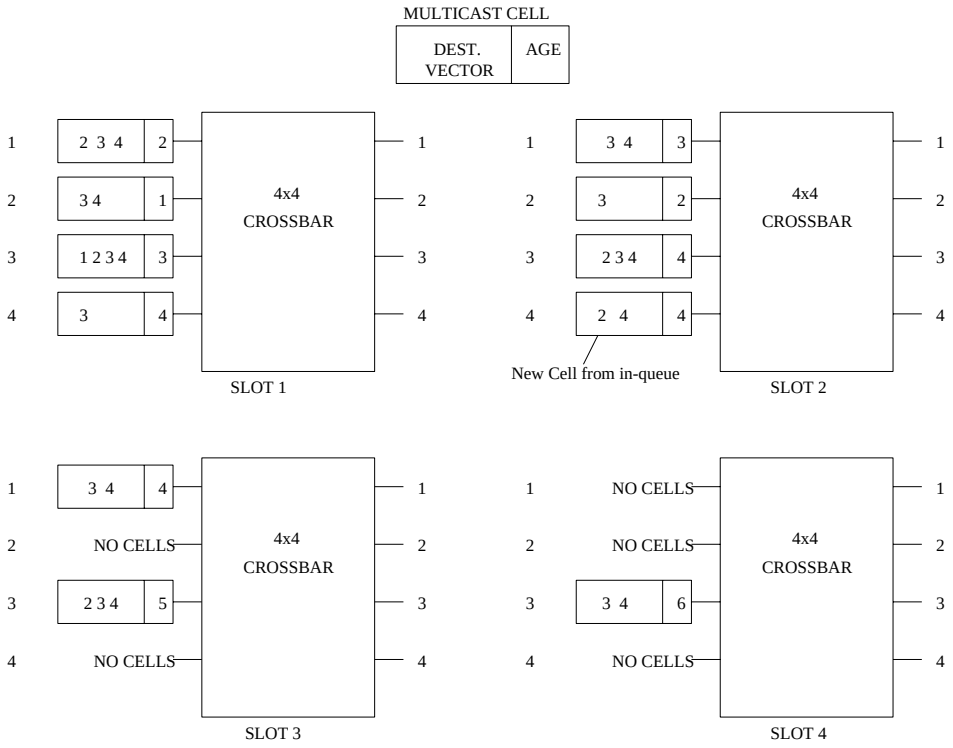

Figure 2.4: Example of WBA on a 4x4 Crossbar Switch. The rectangular blocks indicate multicast cells at the inputs. The first column in the box shows the destination vector and the second column shows the age of the cell.

Figure 2.4 shows the working of WBA on a 4x4 crossbar switch. A weight of 1 is given to the age and a weight of -2 is given to the fan-out. Initially port 4 gets a chance to transmit its cell as it has the maximum weight 2 among all the competing inputs. In the next slot time it injects new cell from its input queue. In SLOT 1 node 3 could reach only one destination 1. The cell is split as two cells, one with destination vector $<1>$ and the other with destination vector $<2,3,4>$. The first cell is transmitted and the second cell remains in the queue and competes with the other cells in the next slot time.

Round Robin Algorithm

The (RSIMPLE) algorithm works by giving preference to one input at a time in a round robin fashion [7]. At each time step t, the multicast scheduler begins by scheduling the cell at the head of input port $(i+1) mod M$, where i denotes the starting port of schedule at $t-1$ and M is the number of input ports. The scheduler continues by examining the head of successive input ports $i+2$, $i+3$. The scheduler does not use fan-out splitting.

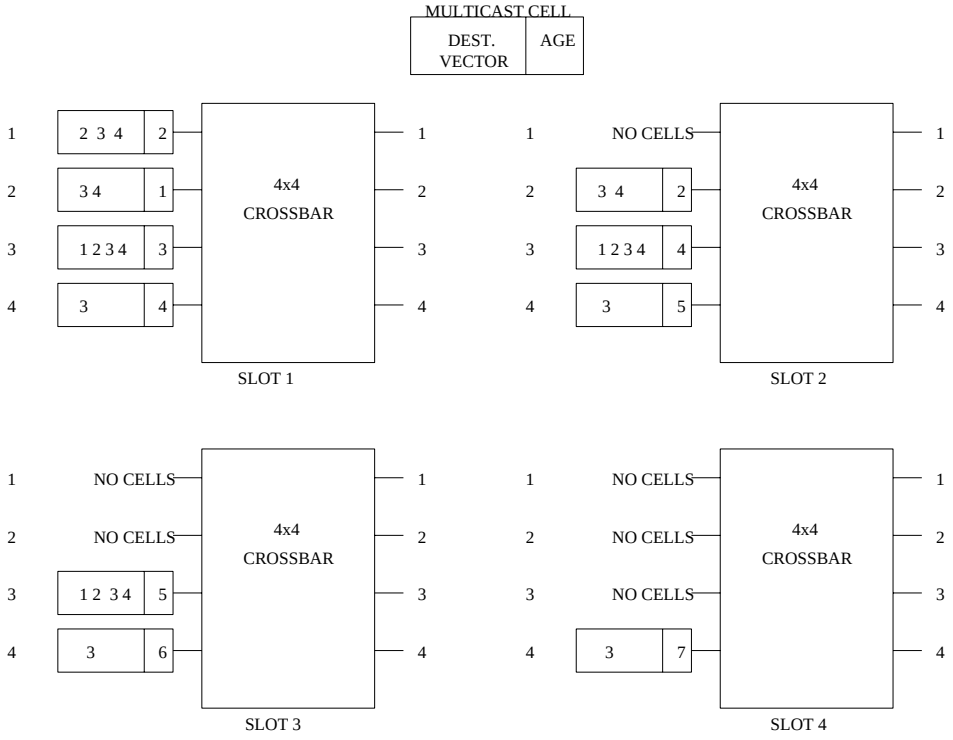

Figure 2.5: Example of Multicast Round Robin on a 4x4 Crossbar Switch. The rectangular blocks indicate multicast cells at the inputs. The first column in the box shows the destination vector and the second column shows the age of the cell.

Figure 2.5 demonstrates the working of round robin scheduling algorithm on a 4x4 crossbar switch. In SLOT 1, port 1 is the prioritized port. It can send the cell at the head of its input queue without splitting. After port 1 chooses <2,3,4> as its desired destinations, no other port can send cells in this slot. All other ports wait for the next slots.

Other Multicast scheduling techniques

We simulated WBA and RSIMPLE and found that using these techniques the switch utilization cannot be increased beyond 70% with correlated arrivals. To increase the multicast performance for input-queued, crossbar-based switches, we experimented with a couple of schemes using a combination of weight based and round robin algorithms. In the first scheme, we used a round robin scheme to give priority to a port in each round. This prioritized port can send a multicast cell without splitting in that slot time. All the other inputs use the WBA scheme to send cells. In the second scheme, a similar round robin scheme is used, but the remaining inputs use a weight based scheme similar to the one used by WBA, except that the weights are given based on the achievable fan-out. Achievable fan-out is the number of output ports (specified in the multicast cell) to which an input port can send the multicast cell without conflicting with the choice made by the prioritized port. Even these techniques do not increase the multicast performance of the input-queued, crossbar-based switches.

2.2 Desirable criteria for multicast switches

We have found, using extensive simulations, that as efficient as a scheduling policy might be, significant improvements in multicast performance cannot be obtained for input-queued, crossbar-based switches. Furthermore, fan-out splitting is necessary, as the performance degrades considerably otherwise. This further increases the complexity of the input-queued switches when used for multicast. The inherent complexity of multicast communication requires high bandwidth inside the switch fabric and flexibility in the number of cells accepted at the output ports. An increase in the switch fabric bandwidth can be obtained either by using multiple copies (or by operating the switch fabric at a higher rate than the line rates) of the switch fabric with output queuing or using buffers inside the switch fabric. Earlier approaches in the design of multicast switches used both these techniques for crossbar and banyan network based switches. Most of the crossbar-based switches for multicast use output queuing with the fabric operating at a higher speed than the line rates [40]. Any attempt to increase the bandwidth of a crossbar based fabric by using multiple copies is not feasible as it increases the cost of the switch

considerably. This approach achieves good multicast performance, but is not scalable in terms of cost. Hence it is not feasible to implement large scale multicast switches using pure output queuing. As an alternative banyan network based switches with both internal and output buffering ([5]) were proposed. Though these switches achieve good multicast performance, they require complex circuitry and are not scalable. From the earlier switch designs and from our observations using extensive simulations of multicast scheduling policies for input-queued, crossbar-based switches, we propose following desirable features for low cost, scalable switches.

- Input queuing is very effective for unicast communication and also it is not feasible to implement large scale switches with pure output queuing. So some kind of buffering at the inputs is desirable.
- Output queuing simplifies multicast scheduling by providing more bandwidth and satisfying the high bandwidth requirements of multicast communication.
- Switch fabric should be simple, inexpensive and fast. This can be achieved by using low-cost switching networks without internal buffering.
- A multicast scheduling policy that does not use fan-out splitting is desirable as it reduces the complexity of the scheduler in input-queued switches.

In the next chapter, we present a new switch design that tries to incorporate most of the features described above (input queuing, output queuing, low cost switch fabric, no fan-out splitting) and yet is cost-effective and simple to implement.

Chapter 3

Ωswitch

3.1 The Design

The Ωswitch is composed of input queues, output queues and multiple copies of banyan network based switch fabric. The switch design is based on the input-queued, banyan network based design proposed for unicast by Boppana et al [1]. To overcome the inherent bandwidth problems with multicast, we have added the output queues to the switch design. The advantage of using output queues along with input queues is that, they allow more than one output cell to be transmitted to the same output in a single time slot, thus increasing the bandwidth inside the switch [33], [34]. (This flexibility provided by output queuing is referred to as increase in switch bandwidth in all further discussions). If pure output queuing is used, each output queue should be able to receive M (MxM switch) cells in the same time slot (We call M the degree of output queuing). The switch has to have a speedup equal to M or has to use M copies of the network. Since this is not feasible for large switches, a combination of input and output queuing is likely to provide a low-cost alternative. These switches would have a speedup less than M(or number of copies less than M) compared to an output-queued switch, providing flexibility in the degree of output queuing. Using a crossbar-based network in these switches would still be expensive as multiple copies of crossbar would result in an increase in the cost of the switch. A banyan-based network would be an alternative for these switches as using multiple copies of banyans will not increase the cost of the switch. Though banyan networks are used, a hardware based selection technique is used

that makes cell selection and routing easier to implement. This eliminates the need for complex network setup and cell selection schemes otherwise needed by these networks.

3.2 Switch Fabric and Selection Network

The switch uses separate FIFO queues (FIFO) for unicast and multicast at each input port. The switch fabric uses two or more copies of the banyan networks. These banyan networks are very much similar to the networks used for unicast, except that replication capabilities are added to the switching elements so that copies of a cell coming in may be delivered to both the outputs. The hardware based cell selection is done by using a selection network that is a copy of the underlying banyan network. The selection network routes only the input connection requests containing the destination vectors in case of multicast cells and a destination port number in case of unicast. Since only the requests (not the cells) are routed through the selection network, multiple iterations can be carried out in a slot time. The selection network selects after one or more iterations, the set of multicast and unicast cells that can be go through the switch fabric without blocking each other. Since the selection technique is hardware based, it is much faster and simplifies the cell selection process. Since the banyan networks are self-routing, once a set of cells are presented (that do not block inside the fabric) the fabric delivers them to the outputs. Since multiple copies of banyans are used in the switch fabric, more than one cell can be received by the output at the same time. (Over a period of time, however, each ouput receives an average of one cell per slot-time). The switch also has an optional cell-splitter. Cell-splitter is used if the multicast scheduling policy implemented uses fan-out splitting.

3.3 Output Queues and Backpressure Mechanism

Output queues shared by both multicasts and unicasts are used to store multiple cells received by the outputs in a slot-time. Each non-empty output queue is drained at the rate of one cell per slot-time. This might result in the overflow of the output queues resulting in high cell losses. To prevent this a backpressure mechanism is used. This mechanism places two thresholds on the output queues, called

the minimum and the maximum thresholds. If any of the output queues size exceeds the minimum threshold limit, it accepts only one cell till enough cells are drained and the queue size falls below the threshold limit. If the size equals the maximum threshold limit, no cells are accepted by that output till enough cells are drained. All the inputs are notified about the queue lengths before the beginning of each slot time. Depending on the queue lengths and the thresholds the inputs either split the cells and send the requests for the new cells or can wait for the next slot time.

We complete the discussion of the switch design by presenting the unicast scheduling on Ωswitch. It is similar to the unicast scheduling done on the switch proposed by Boppana et al., [1], when no multicast traffic is present. Though output queuing is used, at most one unicast cell is scheduled to an output in each slot-time.

3.4 Unicast scheduling on the Ωswitch

The unicast selection process works as follows. In the first iteration, all the inputs with cells to be transmitted send unicast requests with the destination address of the cell being considered through the selection network. The selection network is a copy of the underlying network and routes only destination addresses, not the cells. It examines the destination bits of each request's destination address and depending on the bits, uses either cross-over mode or pass-through mode to route the requests through the network. If there are ties at any stage in the network, they are broken randomly. After the iteration, all the inputs that succeed are notified. In the subsequent iterations, only the inputs not chosen in the earlier round contest with new requests. The inputs that are successful send the requests that won on the earlier rounds and are given priorities when there are ties inside the selection network. After a few iterations, a set of destinations can be found, that can be routed in the next time slot on the actual fabric without conflicts using the paths selected in the selection process.

As an example (Figure 3.1), Consider an 8×8 omega network designed with 2×2 switches. Omega networks are self routing networks. At each stage, a particular bit of the destination address is used to determine how the cell is going to be routed by the 2×2 switch in that stage. For this network, the

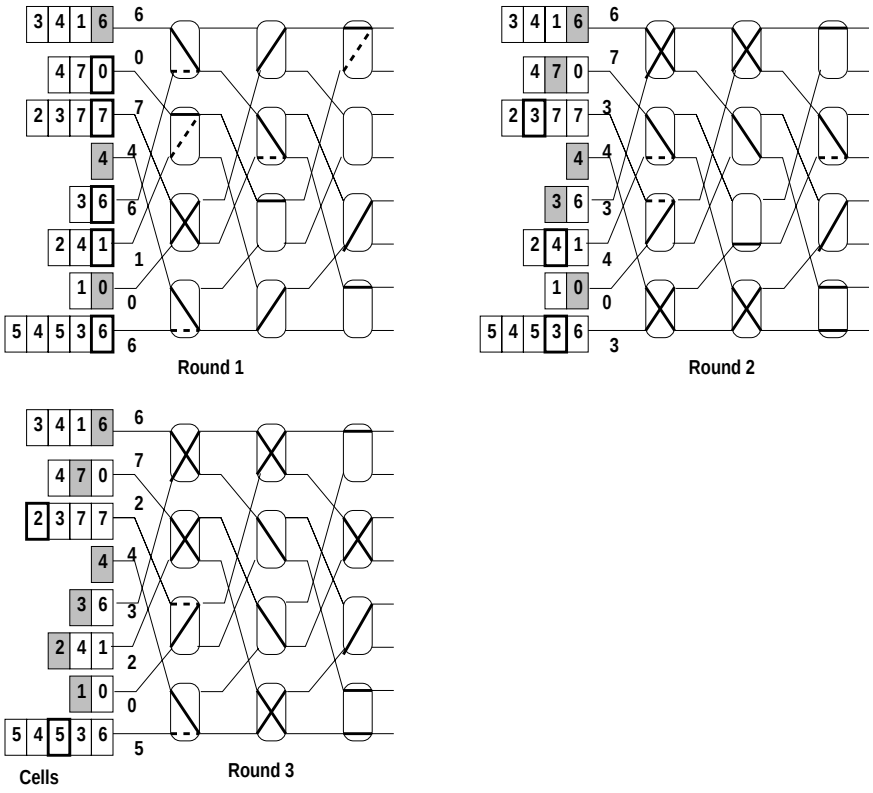

Figure 3.1: Example of Unicast cell selection in the 8x8 Ωswitch. The rectangular blocks indicate the cells (with numbers indicating their destinations) queued at switch inputs. The numbers at the inputs of the first stage of switches indicate the destinations of the cells that inputs wish to send. Shaded cells indicate winning inputs and the cells selected. Thick lined cells indicate inputs with rejected requests. In the network, dashed lines indicate rejected connections and solid lines indicate accepted connections. Inputs with rejected connections select new outputs for future rounds. Inputs with accepted connections use the same outputs for later rounds.

left most bit, middle bit and the right most bit are the routing bits for the left, middle and right stages respectively. If the routing bit of a request is zero, it has to be routed to the upper output of the 2×2 switch, otherwise it is routed to the lower output. If both inputs of a 2×2 switch have the same value for their routing bits then it is called a tie and one of them is dropped. The dropped cell is said to be misrouted and it is not considered in further iterations of that slot time. In Figure 3.1 it can be seen that, three inputs win in the first round, two in the second round and one more in the third round. The paths used by winning outputs are not disturbed in the later rounds.

In the next chapter we present multicast scheduling policies for the Ωswitch in detail.

Chapter 4

Scheduling Multicast cells in the Ωswitch

In this chapter we propose two multicast scheduling policies for the Ωswitch. The first one uses fan-out splitting and uses randomness to prevent starvation and is called random scheduling policy. It has been proved in the past that scheduling policies which use fan-out splitting are work conserving and are necessary to achieve high multicast throughputs for input-queued switches. However, random scheduling policy introduces additional complexity in the selection network of the Ωswitch as cell splitting is used. To simplify the switch design, we considered a simple round robin scheme without cell splitting. Contrary to the earlier observations, this scheduling policy achieves performance comparable to random scheduling policy when only multicast traffic is present. In fact this approach achieves 100% utilization when the Ωswitch is overloaded with unicast and multicast throughput. In the following sections we present these two scheduling algorithms.

4.1 Random Scheduling Policy

The random scheduling policy works like the unicast scheduling policy described earlier, except that here destination address in the requests is a vector. When a multicast request with a destination vector arrives at an input line of a 2×2 switch to be routed, it is split into two vectors. Figure 4.1 shows two multicast requests waiting at a 2×2 switch to be routed through the selection network. There are four possible ways in which the cell can be routed. Figure 4.1 shows all the four cases. Though all four cases are feasible to implement, we use only case 2 (Figure 4.1b) and case 4 (Figure 4.1d) in our simulations.

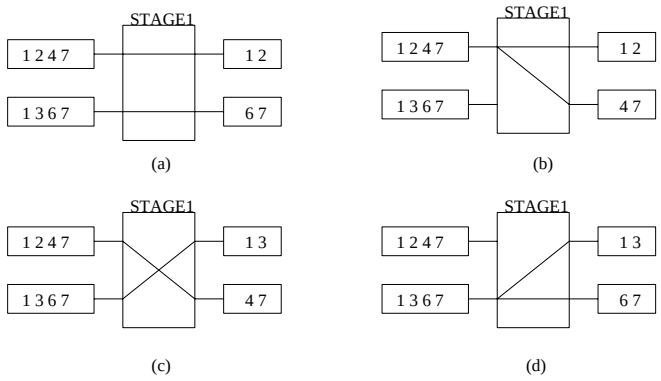

Figure 4.1: Example of splitting of destination vectors by the switching elements of the selection network. (a,b,c,d) represent four different possible ways of splitting. The big rectangular box with stage1 on top, represents the 2x2 switching element. The two boxes at the inputs of switching elements represent the multicast requests with the numbers representing the destination vectors. The boxes at the outputs represent the requests after splitting.

When the destination vector is split into two vectors, the first vector contains all the destinations that have to be routed to the upper output and the second vector contains all the destinations that have to be routed to the lower output. If there are ties from the other input line of the 2×2 switch, they are broken randomly. All the input requests that have passed through the final stage of the selection network are said to be the winning requests. Inputs are notified about the winning requests and the inputs split the cells allowing the winning cells only to go through the actual switching fabric. Since the multicast selection process is time consuming, only one round is allowed through the selection network for each copy of the omega network. Once the outputs are selected by the multicast cells, there are several rounds of unicast selection (section 3.4). Once the paths are fixed, multicast and unicast are allowed through the switch simultaneously. An input can send only one cell, either multicast or unicast, during a time slot.

Before sending a multicast request through the selection network, inputs are notified about the output queue lengths using the back pressure mechanism. Inputs check if any of the output queue lengths are above the threshold limits. If the lengths are above the threshold limits, then the multicast cell is split and the new request, corresponding to new cell, consists of only those destinations whose output queue lengths are below the threshold limits. This further increases the complexity of the Ωswitch.

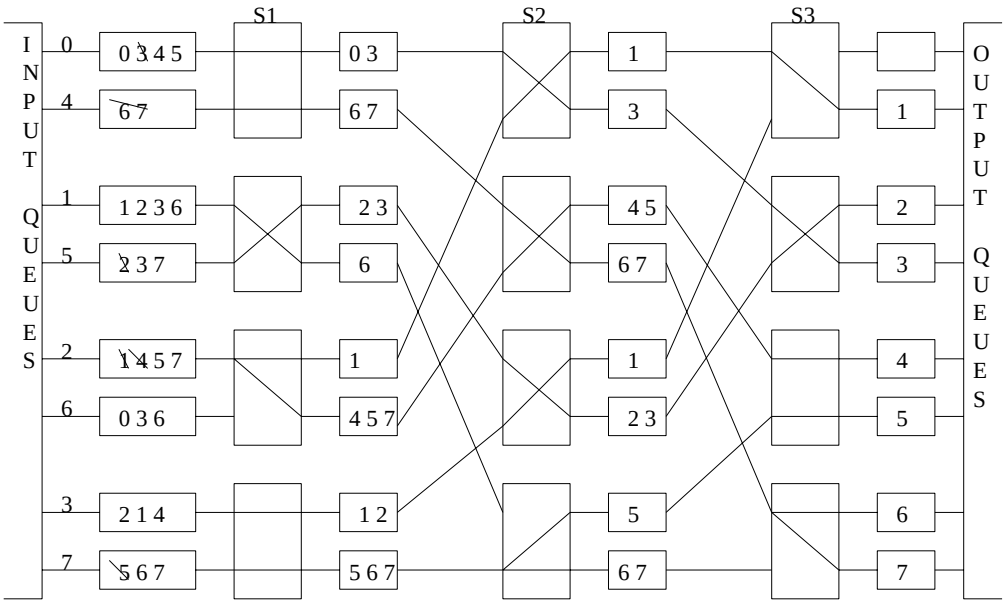

Figure 4.2: Example of Multicast Selection for random scheduling by an 8x8 Ωswitch. The rectangular boxes with numbers in them represent the multicast requests, with numbers representing the destination vectors. Big rectangular boxes with lines in them represent the 2x2 switching elements. The lines represent the paths taken by the requests through the network. Striked out numbers inside the multicast cells, represent the winning destinations after one round of selection.

As an example, Figure 4.2 shows how the selection network operates. The switch shown is an 8×8 switch and is made up of 2×2 switching elements. This example uses all the four cases of splitting shown in Figure 4.1, but in our simulations we use only case 2 (Figure 4.1b) and case 4 (Figure 4.1d). Initially in stage 1, vectors from port 0 and port 4 compete to transmit. Port 0, is successful in sending the vector to the upper output of the 2×2 switch and port 4 is successful in sending to the lower output. Port 0 splits the destination vector and sends only <0,3>. After stage 3, only a few splitted requests (striked out numbers in Figure 4.2) are successful and the appropriate inputs are notified. The inputs then split the cells accordingly. Input 0 splits the cell into two cells, one with destination vector <3> and the other with destination vectors <0, 4, 5>. Ports 5 and port 2 also split the cells accordingly. Port 4 need not split the cells as it can send the cell to all the destinations. There are two such selection rounds for both the copies of the network. Inputs (1,6 and 3), that could not send cells in the first round, compete in the second round for the other copy of the network.

4.2 Round Robin Scheduling Policy

The basic idea of the round robin scheme is to allow two multicast connections (one per copy of the network) without splitting, per slot time. Initially a port i is chosen as a prioritized port. This port i and port $i+1$ are allowed to transmit multicast cells in that slot time. If these inputs have no multicast cells to be transmitted then other inputs are chosen in a round robin manner. That is, if port i has no cell to send then the round robin algorithm starts of from port $i+1$. The first port chosen in this way uses the first copy of the network to transmit its cell and second port chosen uses the second copy of the network. However, in the next slot time the round robin scheme starts from port $i+1$. After the multicast requests of these two cells are allowed through the selection network, unicast requests are scheduled. Since only one multicast is allowed through each copy, there are some paths left in the networks unused. These can be used by the unicast connections. Since no fan-out splitting is required this policy is very easy to implement and also provides fairness for unicast traffic. Inputs can send only one cell per time slot and for multicast cells if one of the destinations output queue limit is above the threshold, then the input does not compete for multicast scheduling. This further simplifies the design of the Ωswitch.

4.3 Complexity

The complexity of the Ωswitch is determined by the scheduling policy used. If the random scheduling scheme is used with fan-out splitting, it needs additional capabilities to be added to the selection networks to chose the multicast cells. Each round of multicast needs some time to route the requests through the selection network and resolve any conflicts that might arise. This increases the complexity of the selection network and also the switch.

If the round robin scheme is used, no fan-out splitting is necessary, since only a multicast connection is allowed per copy of the switch used. In case of a 8×8 switch, since 2 copies are used, it allows two multicasts per slot time. This makes the selection mechanism simpler and faster and leaves enough bandwidth to be used by unicast. Furthermore it does not starve unicast traffic as other schemes might

do. This simple scheme, combined with the input output queued Ωswitch can be a cost-effective solution to handle both multicast and unicast connections efficiently.

Chapter 5

Performance Analysis

In this study, we have proposed variants of existing multicast scheduling techniques for crossbars and a new multicast switch based on multistage networks. To evaluate the performances of various techniques, we have conducted extensive simulations under identical conditions, using a simulator written in CSIM [39] . All the techniques have been simulated with different traffic types, with varying loads of unicast and multicast. The goal of the simulations is to study as to how fast these switches can deliver the cells under different conditions and the percentages of the input load that can be delivered to the output without entailing heavy cell losses at the input queues.

5.1 Simulation Setup

5.1.1 Scheduling algorithms

Input-queued, crossbar-based switches We use WBA [6] and RSIMPLE [7] for multicast scheduling in crossbar-based switches. Though it is well known that RSIMPLE does not give good performance, we present its performance to indicate the baseline multicast performance and the importance of cell-splitting in multicast scheduling for input-queued switches.

Since practical unicast algorithms which achieve close to 100% are already proposed and implemented, we choose a simple exhaustive lookup technique to maximize unicast performance.

The Ωswitch We use random and round robin scheduling techniques for multicast and a network based cell selection (a practical technique but performs slightly worse than the exhaustive lookup used by crossbars) is used for unicast.

5.1.2 Switch Parameters

An 8×8 Ωswitch and an 8×8 input-queued, crossbar-based switch are used for the simulations. Separate queues are used for multicast and unicast at each port. Each queue can buffer up to 128 cells. The minimum threshold for the output queues in case of the Ωswitch is 75%. The selection network of the Ωswitch uses two rounds for multicast (one for each copy) and eight rounds for unicast (four rounds for each copy), per time slot. In all simulations, WBA uses a weight of 1 for the age and -2 for fan-out. A non-empty output queue is drained at the rate of one cell per slot-time.

5.1.3 Traffic types

The scheduling policies for both the switches were simulated under two different arrival processes.

Uncorrelated Arrivals At the beginning of each cell time, a cell arrives at each input with probability p independently of the previous cell arrival and destination.

Correlated Arrivals Cells are generated using a two-state Markov process which alternates between BUSY and IDLE states. The process remains in the busy and the idle states for a geometrically distributed number of cell times, with expected duration $E[B]$ and $E[I]$, respectively. $E[B]$ is fixed at 16 cells for all the simulations. When in this state, cells arrive at the beginning of every cell time and all with the same set of destinations (In case of unicast, to one destination). The arrival rate $p = E[B]/(E[B] + E[I])$.

For both types of traffic, each arriving multicast cell has a destination vector that is uniformly distributed over all possible destination vectors. As a result, for a MxN switch, the average fan-out is $N/2$.

Hence the total load on all inputs combined is $pMN/2$. Since this input load is uniformly distributed on all the outputs, the load is $pM/2$ per output port.

5.1.4 Metrics

Average Delay The time, in cell slot times, it takes for a cell to reach its destination output port from the time it is injected into the input queue. It includes the input queuing delay, transfer time through the switch fabric and in case of Ωswitch it also includes the output queuing delay. It is calculated as the sum of the delay of all the multicast (unicast) cells received, divided by the number of multicast (unicast) cells received.

Overall Delay It is the ratio of the sum of the delay of cells (both multicast and unicast) received, divided by the number of cells (both multicast and unicast) received.

Normalized Throughput or Utilization The number of multicast (unicast) cells received per output port per clock cycle gives the multicast (unicast) utilization. It is calculated by dividing the number of (multicast or unicast) cells received by the product of number of ports and simulation time.

Switch Utilization The number of cells (both multcast and unicast) received at an output port per clock cycle.

5.2 Simulations

A number of simulations have been done with different traffic types and loads, on both the Ωswitch and the crossbar switch. Since a high speed switch may be loaded with only multicast traffic for some portion of time or with only unicast traffic or a combination of unicast and multicast traffic, it has to support all possible traffic combinations. So we have used four types of traffic: Multicast only, Unicast only, Fixed unicast with varying multicast, Fixed multicast with varying unicast. Each simulation was run for 100000 cycles (averaged over 10 runs) with a warmup of 5000 cycles.

Each type of traffic is simulated with both correlated arrivals and uncorrelated arrivals. Hence there are a total of eight possible combinations. The following subsections present the results of the simulations. For mixed traffic, we present the results for only correlated arrivals. We use the following abbreviations for all the plots given below.

WBA - Weight Based Algorithm on the input-queued crossbar switch [6].

RRS - Simple Round Robin on the input-queued crossbar switch [7].

OMS - Random scheduling (section 4.1) with fan-out splitting on the Ωswitch.

OMN - Round Robin (section 4.2) with no fan-out splitting on the Ωswitch.

Uutil - Unicast utilization

Mutil - Multicast utilization

UMutil - Switch Utilization

Udelay - Unicast delay

Mdelay - Multicast delay

UMdelay - Overall delay

The multicast rate is multiplied by $100\times$(average fan-out) to get the percentage multicast load on the switch. For the multicasts we simulated for an 8×8 switch, the average fan-out is 4 (half the number of output ports). A multicast load of 0.25 indicates 100% multicast load on the switch.

5.2.1 Multicast only

This type of traffic simulates the suitability of a switch for multicasts. Figure 5.1 shows the multicast delays and utilization for various techniques, as the multicast rate is varied from 5% to 100%, with uncorrelated arrivals. At around 60% (0.15 multicast rate) multicast load, RRS begins to saturate (delays increase and utilization decreases), as the delays increase and the utilization decreases. RRS saturates earlier than other techniques because it does not use fan-out splitting and hence is not able to sustain the utilization as the input load increases. At around 85% load (0.2125 rate), WBA saturates, as there is not enough bandwidth inside the switch fabric to sustain such heavy loads.

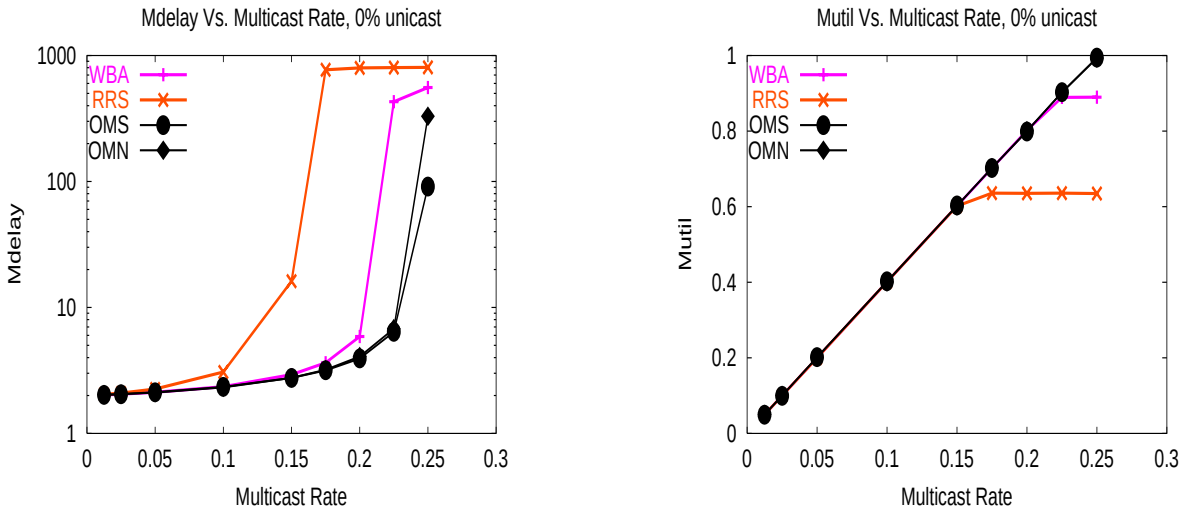

Figure 5.1: Multicast delay and utilization with uncorrelated arrivals.

OMN and OMS saturate in the $95\% - 100\%$ range. OMN does not use fan-out splitting but still performs better than WBA, as there is more bandwidth available inside the fabric. The additional bandwidth is obtained by using two copies of the banyan network with output queuing. OMS gives the best performance of all the techniques. It performs better than OMN, as it uses fan-out splitting, which lets any available output to be used as long as there is traffic.

Figure 5.2 shows the multicast delays and utilization for various techniques, as the multicast rate is varied from 5% to 100%, with correlated arrivals. It can seen (Figure 5.2) that, RRS saturates at around 50% load, WBA saturates at around 70% load, OMS and OMN saturate at around 85% to 90% load outperforming all the schemes. The round robin nature of OMN allows it to fairly schedule multicast cells but the burstiness of the traffic results in an increase in the delays and decrease in the utilization at very high loads. For pure multicast traffic, the Ωswitch is superior to input-queued crossbar switches using the best of the multicast scheduling policies. OMS gives the best performance in terms of delay. On the otherhand OMN has a much simpler multicast scheduling policy that makes it an attractive design. Splitting is a must if good performance is to be achieved with limited switch fabric capacity.

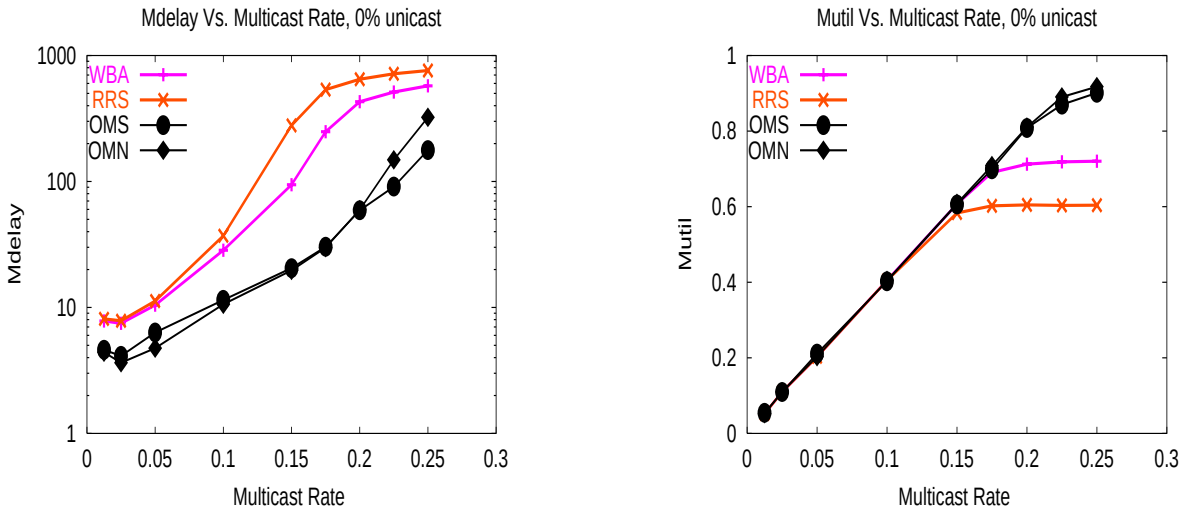

Figure 5.2: Multicast delay and utilization with correlated arrivals.

5.2.2 Unicast only

The unicast only traffic is simulated to evaluate the performance of the switch when only unicast traffic is present. Figure 5.3 shows the unicast delays and utilization for various techniques, as the unicast rate is varied from 10% to 100%, with uncorrelated arrivals. Figure 5.4 gives the same information for correlated arrivals.

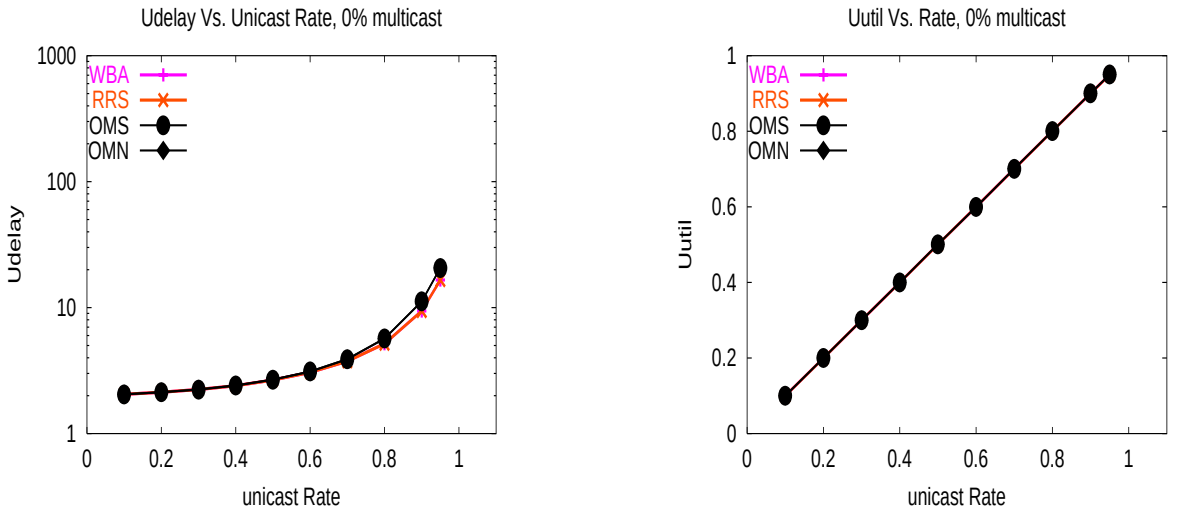

Figure 5.3: Unicast delay and utilization with uncorrelated arrivals.

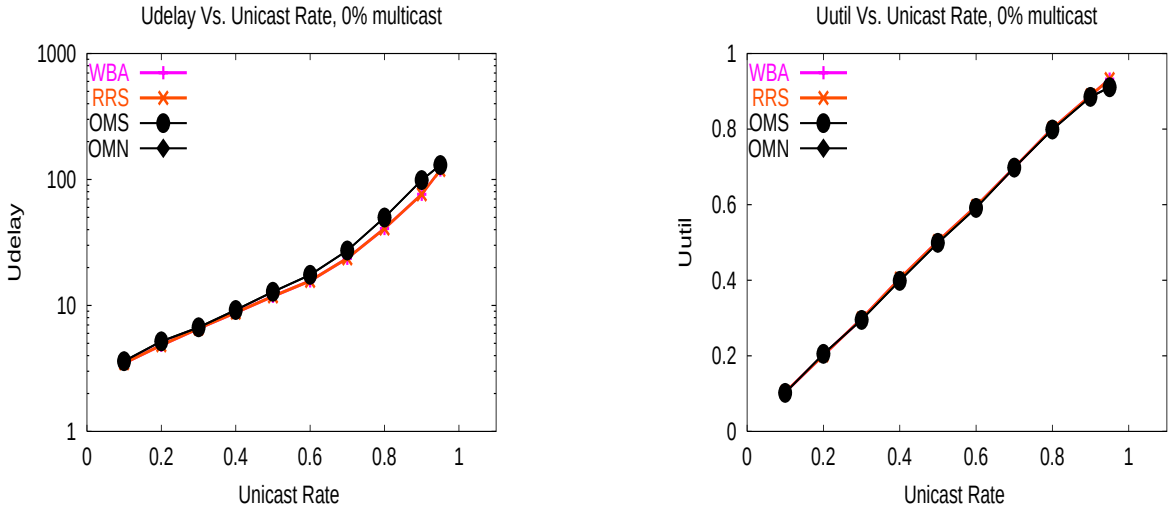

Figure 5.4: Unicast delay and utilization with correlated arrivals.

The Ωswitch performs as well as any crossbar based scheme for uncorrelated arrivals. All techniques sustain 100% load. For correlated arrivals, Ωswitch schemes perform slightly worse than the crossbar based scheme for loads above 80%, as two unicast cells are not allowed for the same output though output queuing is used.

5.2.3 Fixed Unicast, Varying Multicast

This section presents the results of the simulations done with fixed unicast loads and varying multicast loads. These type of simulations help to study the performance of the switch, when a certain percentage of the load on the switch is fixed for unicast and the multicast load is allowed to vary. Several sets of simulations are run by fixing the unicast load at 25%, 50%, 75% and 100%, and by varying the multicast load from 5% to 100%. All simulations are run with both correlated arrivals and uncorrelated arrivals (we present only correlated arrivals case). For all techniques, multicast cells are given higher priority over unicast cells, i.e., during a time slot, first multicast cells are scheduled followed by unicast cells. During a slot time, unicast cells take the paths in the switch fabric not taken by the multicast cells. So multicast cells are never delayed due to unicast cells waiting at input queues. In effect there is a chance that at high multicast loads unicast connections can be starved.

We study the performance of these switches under heavily loaded conditions. When a combination of unicast and multicast traffic is present, the switch may be loaded beyond 100% of its capacity. The following graphs show the delay and utilization for both the switches, with correlated arrivals. First we present the delay graphs for all the four fixed unicast loads, followed by the utilization graphs.

Delay Analysis

There are three graphs (multicast delay, unicast delay and overall delay) for each one of the fixed unciast loads.

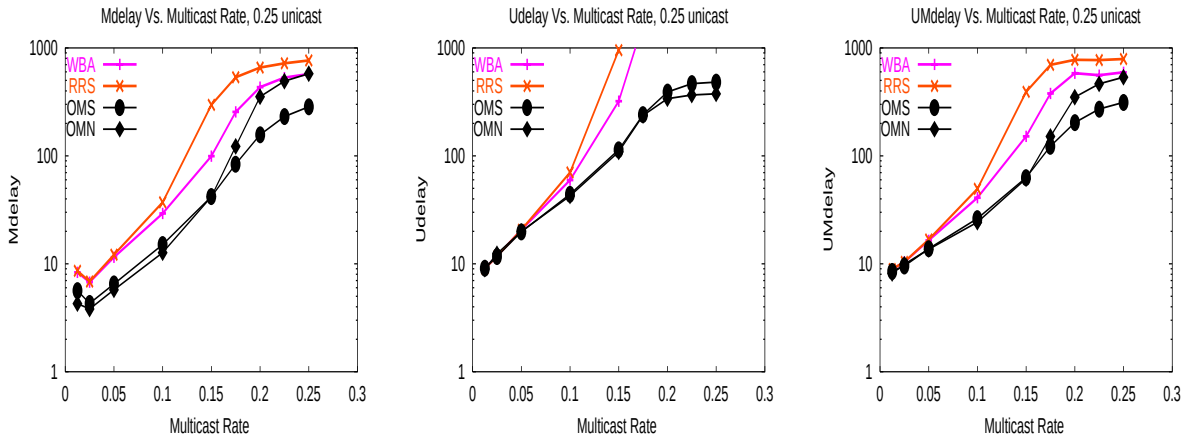

Figure 5.5: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 25% unicast load.

From Figure 5.5, under low unicast loads both OMN and OMS outperform WBA. Though for low multicast loads ($<50\%$), the multicast delays of OMN and OMS are almost the same as the multicast load increases, multicast delays of OMN start to increase. This is because, OMN does not use fan-out splitting and allows considerable unicast through the switch. Once the unicast reaches the output queues, it competes with multicast cells for buffer space. Since multicast delays also include output queue delays, it increases the delay of multicast cells for OMN. Though OMS also allows unicast through the switch, the use of fan-out splitting reduces the bandwidth inside the switch for unicast cells compared to OMN. Hence the unicast delays are higher for OMS than OMN. As the unicast load is increased to 50% (Figure 5.6), the multicast performance of OMN matches that of WBA. Again OMS outperforms

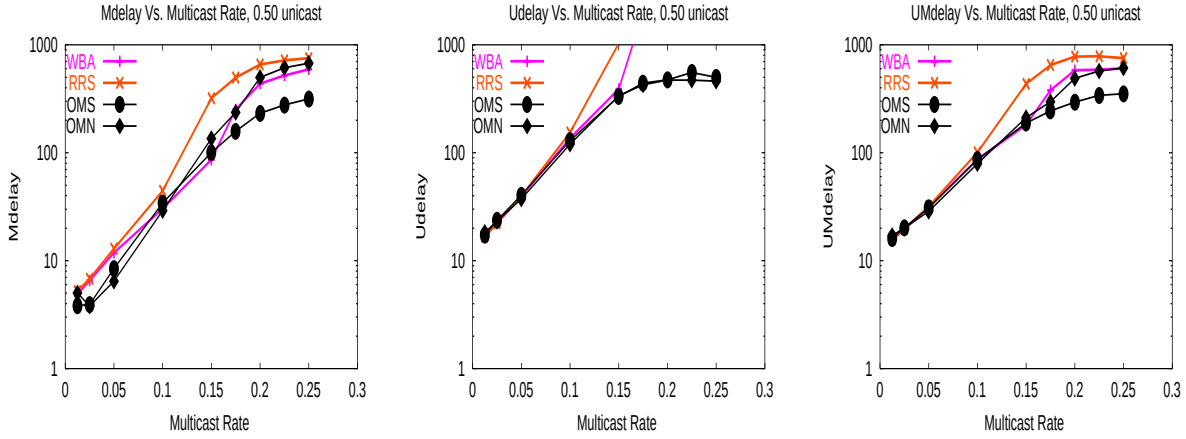

Figure 5.6: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 50% unicast load.

all the other techniques. As the multicast load increases, WBA starves unicast and the unicast delays increase considerably. Though OMS and OMN give preference to multicast, the use of output queuing and multiple copies of switching network provide additional bandwidth inside the switch.

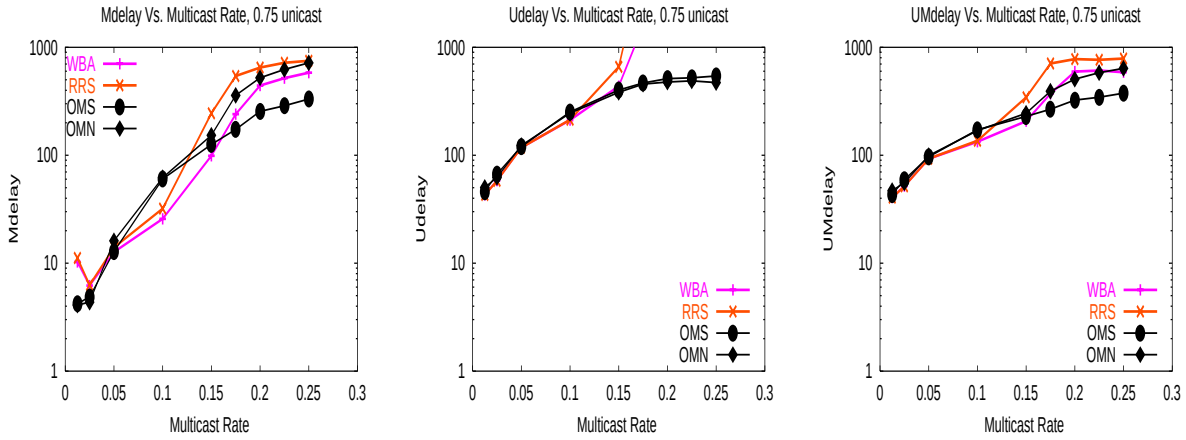

Figure 5.7: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 75% unicast load.

As the unicast load is increased to 75% and 100% (Figure 5.7 and 5.8), there are significant changes in the performance of these algorithms. The multicast delays of OMN and OMS increase considerably. In fact, OMN performs worse than WBA. when the multicast load is below 70% the multicast delays of OMS are also higher than that of WBA.Though high unicast load effects the multicast performance of OMS and OMN, they still have lesser unicast and overall delays.

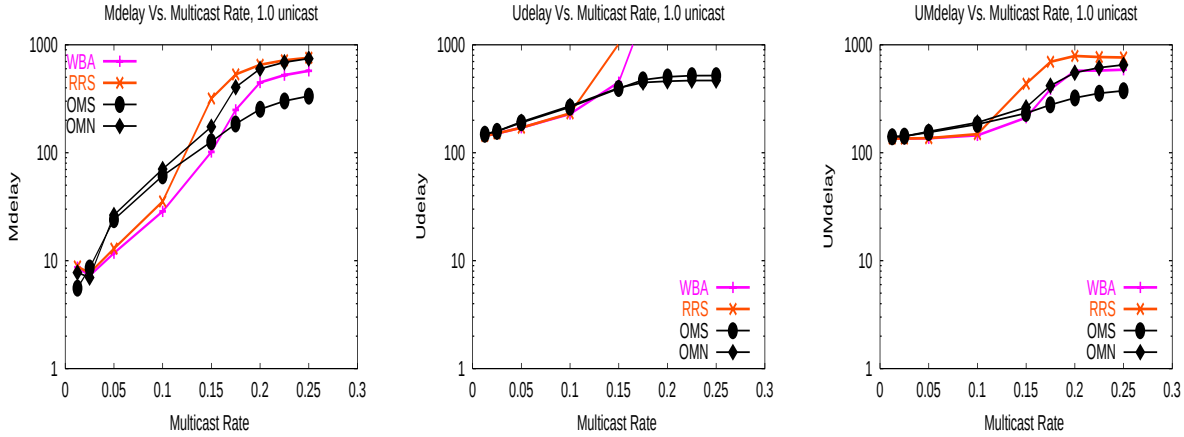

Figure 5.8: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 100% unicast load.

Since OMN uses no fan-out splitting, there are only two multicasts scheduled per time slot. The remaining bandwidth in the switch can be used by the unicasts. The high multicast delays for Ωswitch can be explained as follows: The multicast delays of WBA remain the same as the unicast load is increased, because multicast cells take all the available bandwidth using cell splitting and unicast cells are starved. In Ωswitch, the output queues are shared by both unicast and multicast cells. Though at the inputs, preference is given to multicast, once a unicast cell is scheduled, it competes for resources at the outputs with the multicast cells. Since delay for the Ωswitch also includes output queue delays, the multicast delays increase. When the output queue is full, the back pressure mechanism prevents any cells from being sent. Hence the multicast delays increase.

Utilization Analysis

There are three graphs (multicast utilization, unicast utilization and switch utilization) for each one of the fixed unicast loads.

As the unicast load is increased from 25% to 100%, OMS and OMN try to accommodate more unicast cells resulting in a lower multicast utilization and higher unicast utilization. The backpressure mechanism prevents multicast cells from being sent by OMN, if any of its destinations output queue length is equal to maximum threshold. This results in a further decrease in OMN's multicast utilization

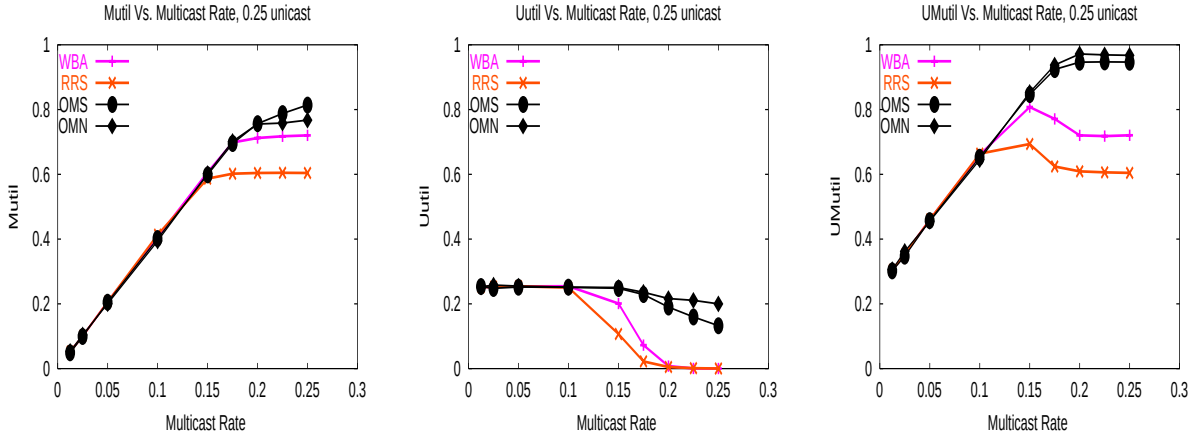

Figure 5.9: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 25% unicast load.

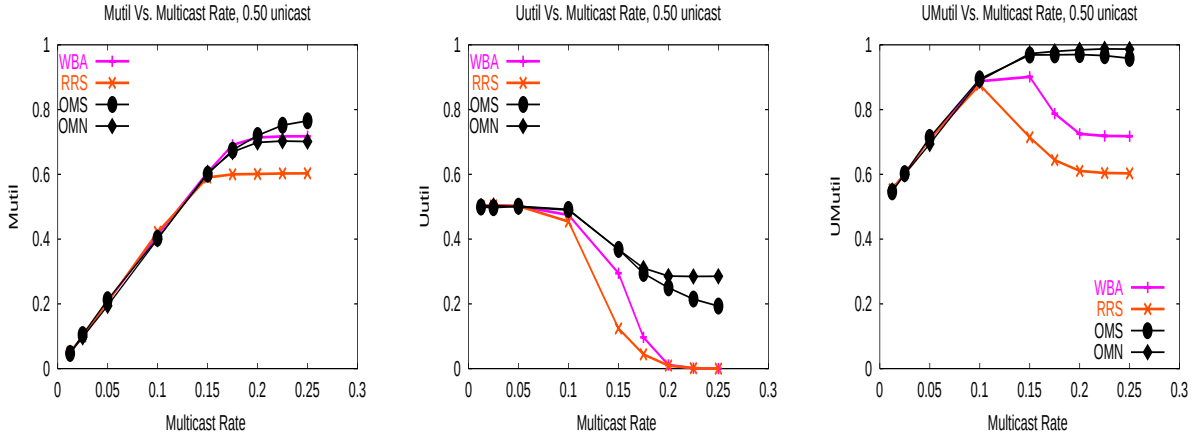

Figure 5.10: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 50% unicast load.

and increase in unicast utilization. Hence OMS gives the best unicast performance and OMN performs as well as the crossbar based schemes for multicast traffic. Though WBA gives slightly better multicast utilization compared to OMN, this is not desirable as at high loads it starves unicast completely. When both the switches are over loaded, OMN tries to achieve a switch utilization of 100%, while OMS achieves a switch utilization of 95%. It can be seen that WBA achieves a switch utilization of only 75%.

The performance of OMN is particularly noteworthy. Under heavily loaded conditions, it schedules both multicast and unicast efficiently, and achieves nearly 100% switch utilization. This performance advantage combined with its simplicity make it an attractive scheme to use. For OMS, when the output

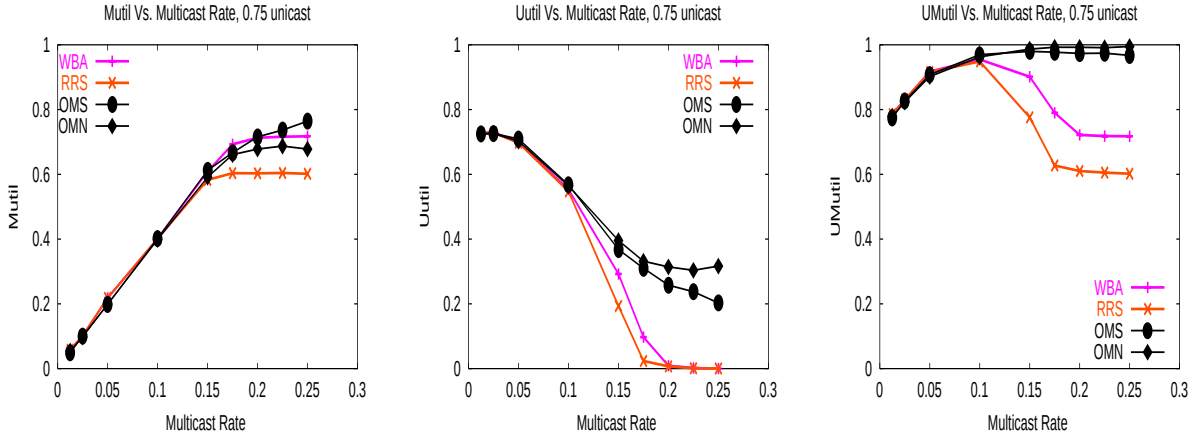

Figure 5.11: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 75% unicast load.

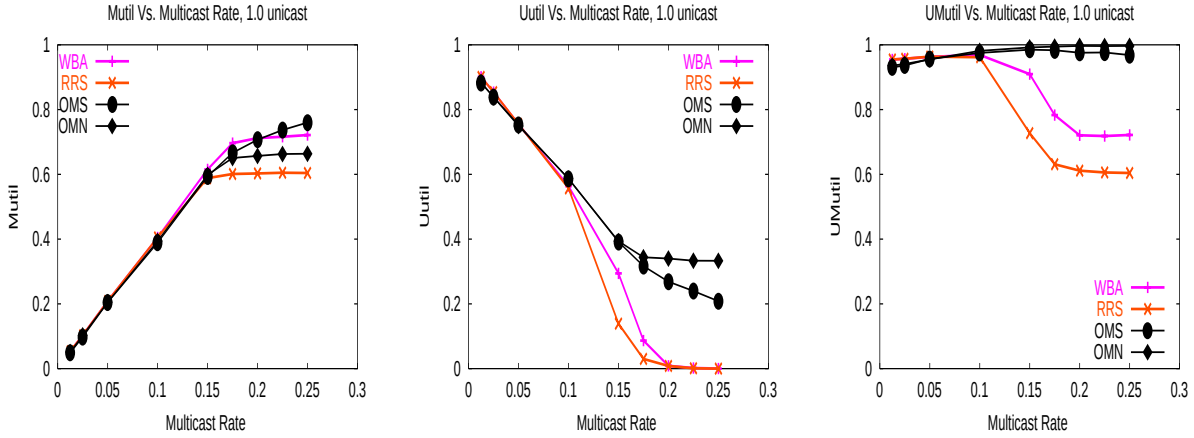

Figure 5.12: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 100% unicast load.

queue capacity reaches the threshold limit, inputs examine the multicast cells at the head of the queue. Inputs split the multicast cells if some of the destinations output queue lengths in the cells destination vector, are more than or equal to the threshold limits. This gives it a considerable advantage over OMN in terms of greater multicast utilization. But, for OMN, since no splitting is used, if any of the destinations specified by the destination vectors output queue lengths are full, then that multicast cell is not scheduled. This gives more bandwidth for the unicast connections. Hence the switch utilization is higher with OMN than with OMS.

5.2.4 Fixed Multicast, Varying Unicast

This section presents the results of the simulations done with fixed multicast and varying unicast loads. Several sets of simulations are run by fixing the multicast load at 25%, 50%, 75% and 100% and by varying the unicast load from 10% to 100%. All the simulations are run with both correlated arrivals and uncorrelated arrivals, though we present only correlated arrivals. In all the simulations, no specific preference is given to unicast cells. During a slot time, unicast cells take the paths in the switch fabric not taken by the multicast cells. We study the performance of these switches under heavily loaded conditions. When a combination of unicast and multicast traffic is present, the switch may be loaded beyond 100%.

The following graphs show the delay and utilization for both the switches, with correlated arrivals. First we present the delay graphs, followed by the utilization graphs.

Delay Analysis

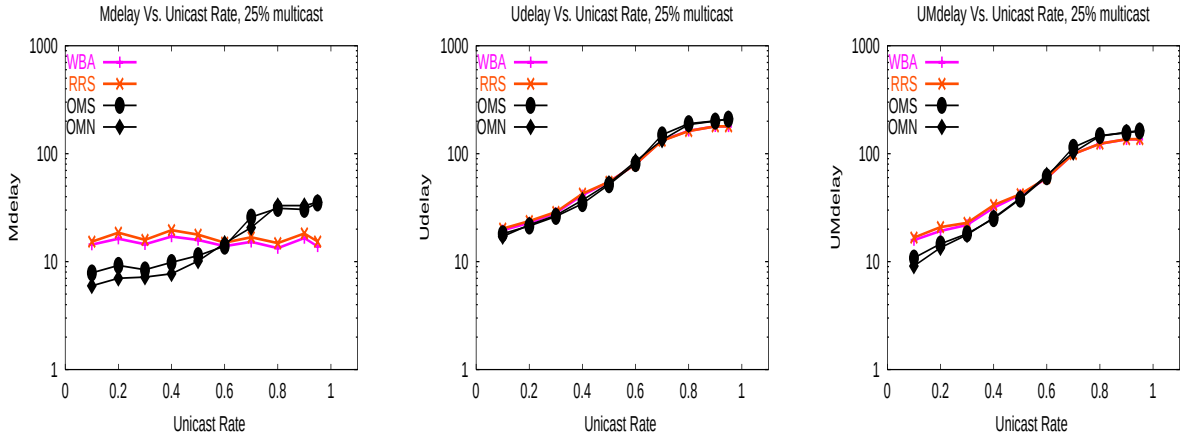

Figure 5.13: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 25% multicast load.

From the graphs (Figures 5.13 - 5.16), it can be seen that, under low multicast load (25%), WBA performs better than OMS and OMN as the unicast load increases. This is because, the unicast algorithm used on the Ωswitch performs worse than the unicast algorithm used on crossbar switches. The advantages of output queuing are limited only to multicasts. But as the multicast load increases, the

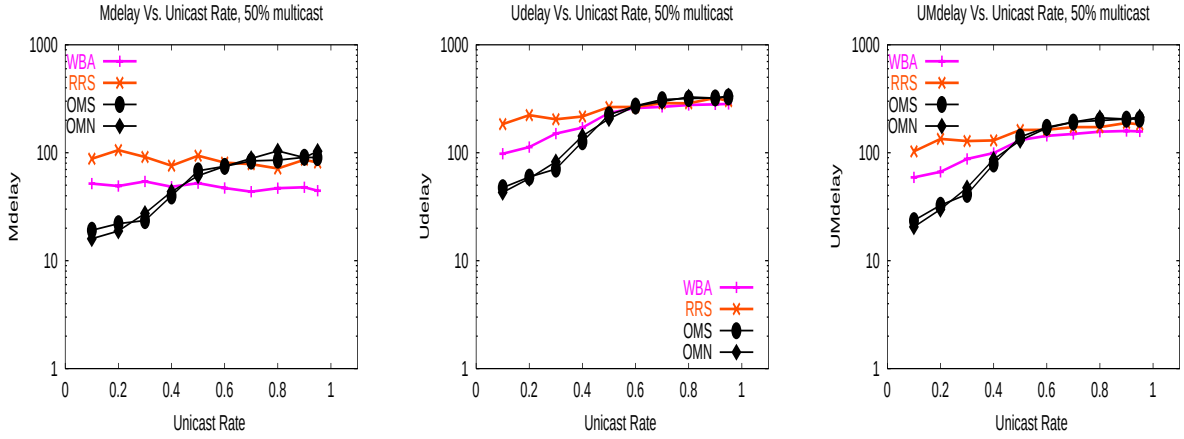

Figure 5.14: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 50% multicast load.

multicast delays of WBA and OMN increase considerably.

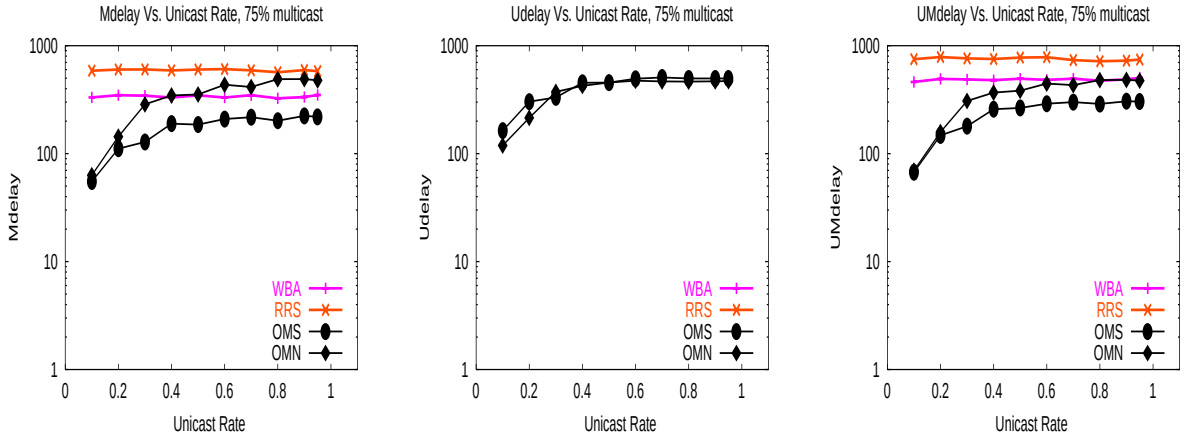

Figure 5.15: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 75% multicast load. Unicast delays for WBA and RRS are very high and are not shown in the Udelay graph.

Only OMS is able to sustain such heavy loads. Again OMN gives the best unicast performance. The overall performance of OMS is better than that of OMN. Again the same explanation can be given, WBA cannot sustain heavy multicast loads (beyond 70%), hence when the multicast load is increased beyond 70%, they cannot match OMS in both multicast and unicast performance. The use of shared output queues for both unicast and multicast effects the multicast delays of OMN more than OMS and results in higher multicast delays for OMN.

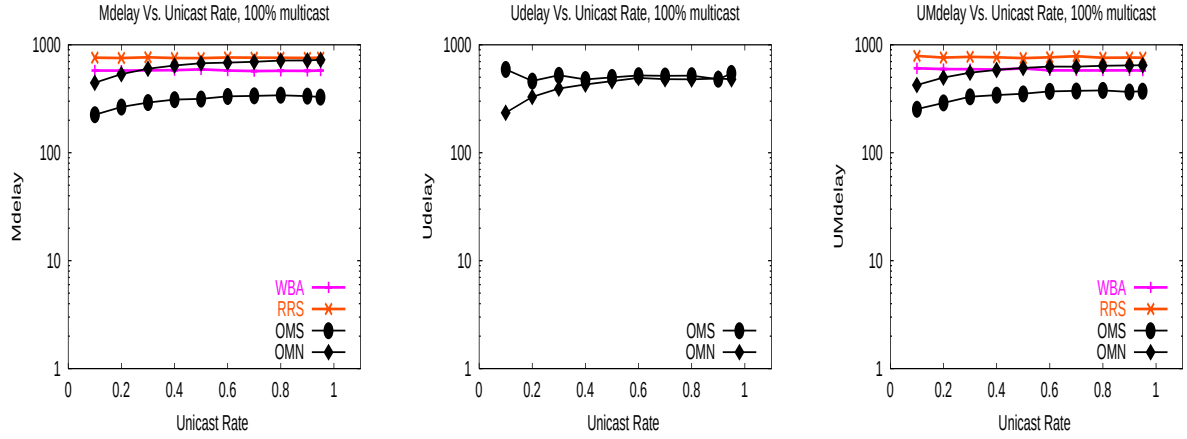

Figure 5.16: Multicast delay, Unicast delay and Overall delay with correlated arrivals, 100% multicast load. Unicast delays for WBA and RRS are very high and are not shown in the Udelay graph.

Utilization Analysis

There are three graphs (multicast utilization, unicast utilization and switch utilization) for each one of the fixed unciast loads.

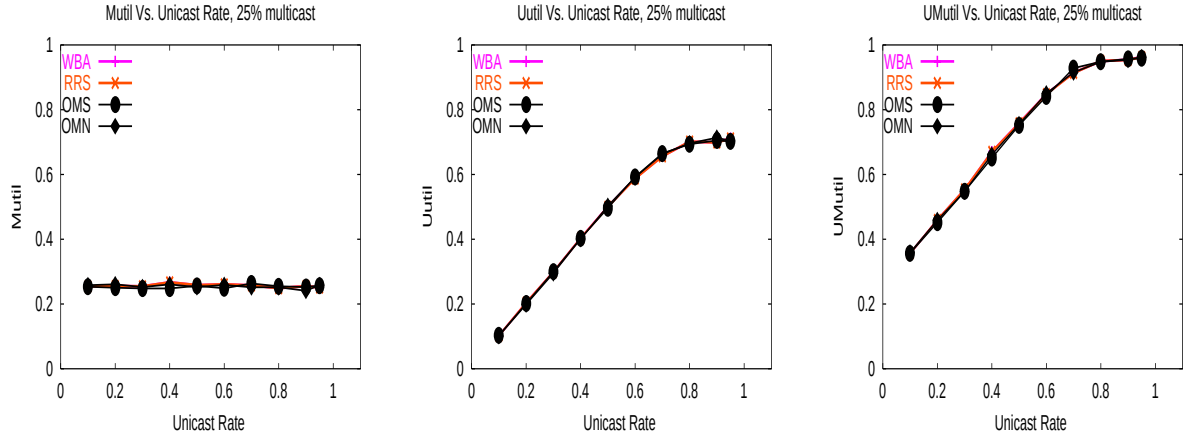

Figure 5.17: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 25% multicast load.

From the graphs (Figure 5.17 - 5.20) it can be seen that under low multicast loads (25%), all the schemes provide the same multicast and unicast utilization. But, as the multicast load is increased to 50%, the unicast utilization of WBA begins to decrease and the unicast utilization and switch utilization of OMS and OMN are higher than WBA.

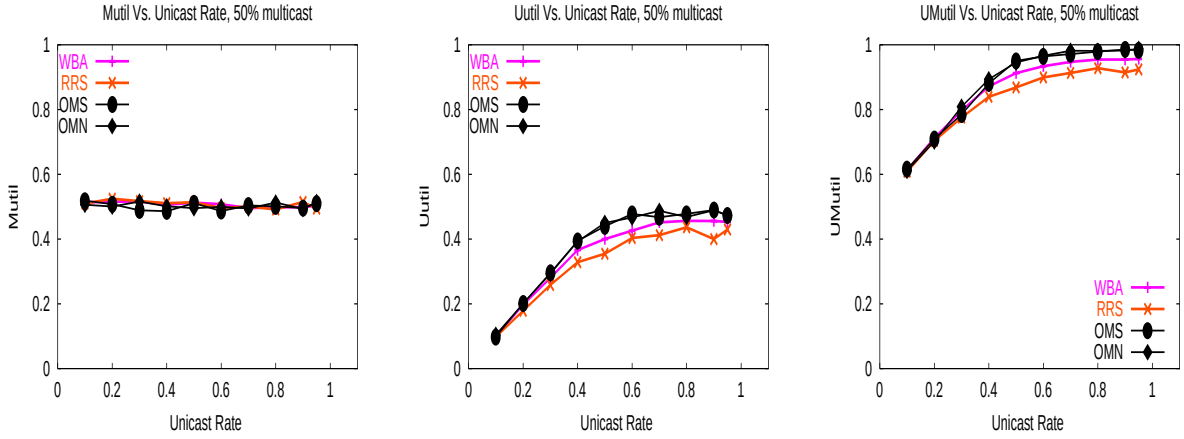

Figure 5.18: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 50% multicast load.

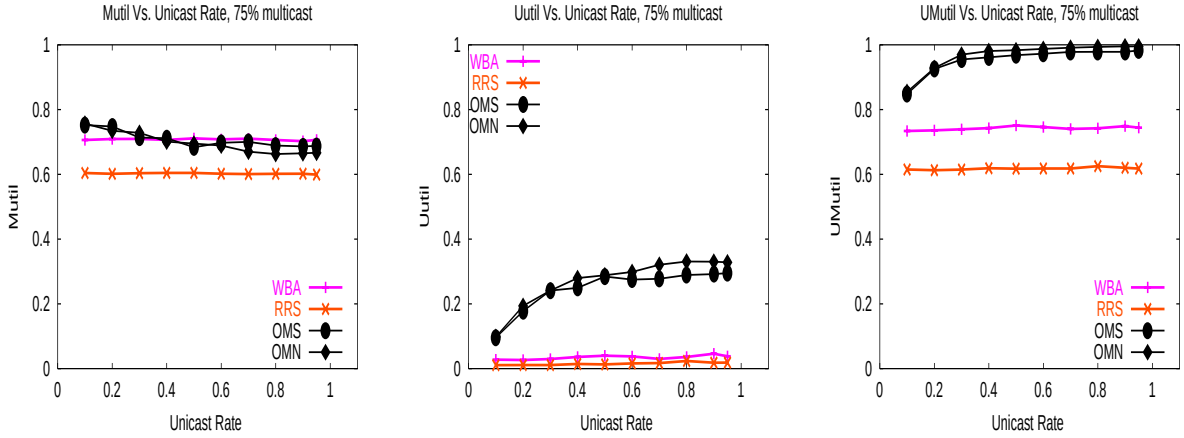

Figure 5.19: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 75% multicast load.

As the multicast load is increased to 75% WBA achieves better multicast utilization than OMN, but this is not desirable as it starves the unicast connections completely. OMN gives better unicast utilization and at high unicast loads achieves 100% switch utilization. As the multicast load is increased to 100%, OMS gives the best multicast utilization. As expected WBA gives around 70% utilization which is the maximum utilization it can give. OMN gives the best unicast utilization and gives close to 100% switch utilization followed by OMS at 95%. Thus it can be seen that specially at high unicast and multicast loads, OMN tries to allow significant amount of unicast through the switch and achieves close to 100% switch utilization.

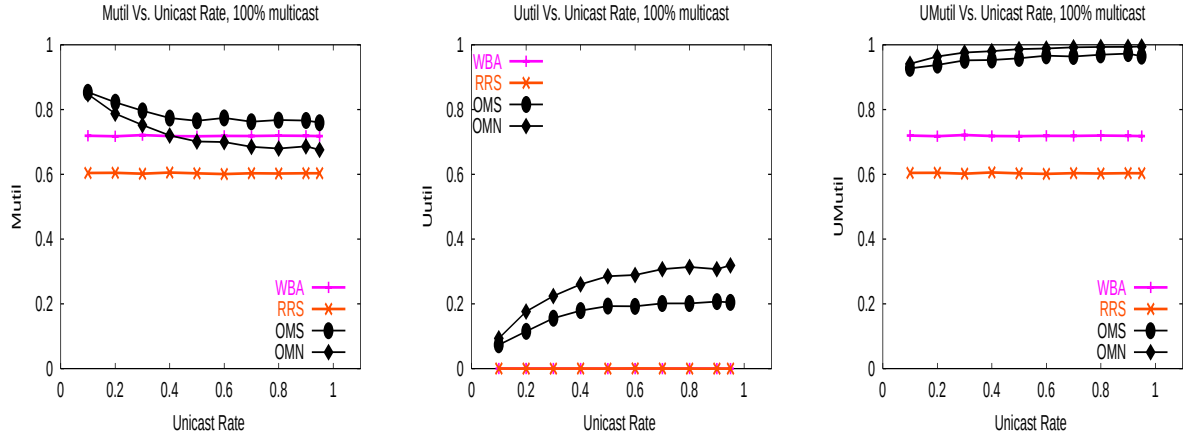

Figure 5.20: Multicast utilization, Unicast utilization and Switch utilization, with correlated arrivals and 100% multicast load.

5.2.5 Summary

When the switch is loaded with only multicast traffic, both the Ωswitch designs OMS and OMN, outperform input-queued, crossbar-based switch designs WBA and RRS. Round Robin Scheme without splitting (RRS), cannot sustain a load of more than 50%. This indicates that splitting is necessary in an input-queued switch. Even with splitting, input-queued schemes cannot achieve switch utilization of more than 70%. However, OMN outperforms the input-queued schemes achieving a utilization of 90% without using fan-out splitting. Though the delays of OMN are slightly higher than that of OMS at high multicast loads, the low complexity and slightly higher utilization of OMN at high multicast loads makes OMN an ideal choice for multicast scheduling on the Ωswitch. OMS gives the best multicast delays of all the four schemes. Hence for good multicast performance fan-out splitting is necessary.

When the switch is loaded with only uncorrelated unicast traffic, all four designs perform similarly. For correlated unicast traffic, crossbar-based designs perform better than the Ωswitch designs. This is because, Ωswitch does not use any of the benefits of output queuing to schedule unicast traffic and hence the unicast on the Ωswitch is scheduled as in any other input-queued switch. This limits two unicasts from being scheduled to the same output in a single slot time. Though two copies of the switching network are used, the total number of point to point connections available during any slot time can

at best match the crossbar-based schemes. But at high loads, the blocking nature of banyan networks reduces the unicast utilization and increases the unicast delays. Hence crossbar-based schemes perform better than the Ωswitch based schemes under high unicast loads.

When the switch is loaded with fixed unicast varying correlated multicast traffic, as the unicast load is increased from 25% to 100%, the multicast performances of WBA and RRS remain the same. Both WBA and RRS give preference to multicast. As the multicast load increases, all the available bandwidth in the switch is used up by multicast connections. If there are any paths left in the switch fabric not used by the multicast cells, unicast cells are scheduled. At high loads if all paths are used up by multicast cells then unicast cells are starved. Hence the presence of unicast cells does not effect the multicast performance of WBA and RRS. But this adversely effects the unicast performance, and as a result, WBA can achieve a maximum switch utilization of 70% only. On the otherhand, multicast delays of OMN and OMS increase with increase in the unicast load. OMN and OMS give preference to multicast cells over unicast cells. But even at high multicast loads there is enough bandwidth inside the switch that is provided by multiple banyan network copies and output queuing, to schedule some unicast cells. Once these unicast cells reach the outputs, they share the same output queue with multicast cells, increasing the output queue delays. Since the multicast delays for the Ωswitch also include output queue delays, the multicast delays of both OMN and OMS increase as more and more unicasts are scheduled. The multicast performance of OMN and OMS is effected by unicast traffic, but the slight decrease in multicast performance is offset by better unicast utilization. Under high traffic conditions, OMN allows significant amount of unicast through the switch and achieves switch utilization of close to 100% while OMS achieves close to 95% switch utilization.

when the switch is loaded with fixed multicast and varying unicast traffic (correlated arrivals), at multicast rates below 70% (at this point, WBA saturates), WBAs multicast performance matches that of OMS and OMN. This is because, WBAs performance does not change with increasing unicasts, as no preference is given to unicasts. The performance of OMN and OMS decreases slightly with increasing unicasts, as unicast cells share the same output queue with multicast cells, increasing the

multicast delays. Hence the multicast delays of OMN and OMS increase and match the multicast delays of WBA. This decrease in multicast performance is offset by increase in unicast performance for OMN and OMS. But, as the multicast loads are increased to 75% and 100%, WBA cannot sustain heavy loads and the delays increase considerably and the utilization decreases. At these high multicast loads, WBA does not schedule unicast cells and starves unicast connections completely while OMS and OMN give better performance, allowing a significant amount of unicast load through the switch. Again, OMS outperforms all the schemes in terms of multicast delays at high loads and OMN gives the best unicast and overall performance.

Both OMN and OMS outperform the crossbar-based schemes, but there are some performance differences between these two schemes. When only pure multicast traffic is present, OMS gives better delay performance as using splitting it can schedule cells faster. But OMN gives slightly better utilization, showing that the round robin scheme can handle burstiness efficiently by delivering more cells. When a combination of unicast and multicast traffic is present, multicast performance of OMS is better than that of OMN. This is because, OMN uses no fan-out splitting, hence schedules a maximum of two multicasts per slot time. This leaves enough bandwidth inside the switch for OMN allowing it to schedule more number of unicasts than OMS. The resulting backpressure further limits multicast cells from being sent, resulting in a decrease in the multicast utilization. since this decrease is offset by an increase in the unicast utilization and decrease in unicast delays, OMN is able to schedule unicasts and multicasts efficiently and at high loads achieves a switch utilization of close to 100%, while OMS achieves 95% utilization. If multicast communication is of greater importance, the decrease in multicast performance of OMN with increase of unicast loads can be prevented by using separate output queues for multicasts and unicasts. Though OMS achieves better multicast performance, it requires cell splitting, that needs additional hardware. This complicates the Ωswitch design as the number of ports increases. Hence OMN may be a more attractive choice for Ωswitch.

Chapter 6

Conclusions

6.1 Conclusions

The increase in demand for network bandwidth creates a need for high-speed multicast switches. The high bandwidth requirement of multicast, requires these switches to provide high bandwidth inside the switch fabric. Any attempt to increase the bandwidth effects the scalability of the switch. To provide a cost-effective and scalable design. we present the design of a high-speed multicast switch called the Ωswitch, that is based on the banyan switch fabrics. The Ωswitch uses both input and output queuing, a hardware based cell selection policy, a backpressure mechanism to prevent output queue losses, and two copies of the banyan network as switch fabric. Since banyan networks are much cheaper compared to crossbars the switch design is cost-effective.

We also present two scheduling policies, random (OMS) and round Robin (OMN) for multicast scheduling on the Ωswitch. random scheduling uses fan-out splitting and requires a complex selection process to schedule the multicast cells. OMN allows only one multicast cell through each copy of the banyan network per time slot and does not use fan-out splitting. OMN is not only simple but also allows significant amount of unicast to go through the switch and achieves 100% utilization at very high loads of combined unicast and multicast traffic. A simple round robin scheme with a look ahead into the input queues was used for unicast scheduling on the crossbar switch, and a network hardware specific selection technique proposed in [1] was used for unicast scheduling on the Ωswitch. We also studied

existing multicast scheduling techniques, WBA and RRS, for input-queued, crossbar-based switches and proposed extensions to them. Extensive simulations were conducted to study the performance of both the switches operating different scheduling policies and with varying unicast and multicast loads.

The simulation results show that with pure multicast traffic, the Ωswitch based schemes, OMS and OMN, outperform input-queued, crossbar-based schemes, WBA and RRS. With correlated multicast traffic WBA can achieve only 70% utilization while OMN and OMS achieve close to 90% utilization. This clearly indicates that input-queued switches are not suitable for multicasting as they provide limited bandwidth inside the switch. OMN achieves slightly better utilization compared to OMS indicating the effectiveness of round robin algorithm in handling bursty traffic. However OMS gives the best delay performance of all the schemes indicating the effectiveness of splitting in delivering multicast cells quickly.

With pure unicast traffic, crossbar-based switch designs give better performance than Ωswitch at high unicast loads. This is because in an Ωswitch, an output cannot receive more than one unicast cell in a single time slot. The advantages of output queuing are only limited to traffic containing multicasts. Unicast scheduling in Ωswitch is handled using input queues only. Hence the blocking nature of the Ωswitch fabric effects the unicast performance at high loads.

We also have studied the performance of Ωswitch with a combination of unicast and multicast traffic. The multicast performance of WBA remains the same with increase in the unicast traffic. This is because, WBA gives preference to multicast and if any bandwidth is left in the switch fabric it is used by unicasts. OMN and OMS also give preference to multicasts, but once a unicast cell reaches an output port it competes with the multicast cells for the output queue buffer space, which is shared by unicasts and multicasts. Since the multicast delays for OMN and OMS include the output queue delays, the multicast delays increase with an increase in the unicast traffic. This problem is more severe in OMN compared to OMS. However, OMN schedules more unicast cells than OMS and achieves close to 100% switch utilization while OMS achieves close to 95% and WBA achieves only 70%.

Hence we conclude that Ωswitch with simple round robin multicast scheduling policy without fan-out splitting can be an attractive choice for multicasting.

6.2 Further Work

The proposed Ωswitch uses two copies of the banyan networks as the switch fabric. The number of copies can be increased to three or more. This provides additional bandwidth inside the switch fabric but increases the complexity of the design of the output queues. In the performance analysis we have used 8×8 switches. This analysis can also be extended for large switches with more number of ports. As the number of ports increases, for round robin scheme to perform as well as the schemes which use splitting, it might be needed to use more than two copies of the network. The round robin algorithm can also be slightly modified to search for additional multicasts which can be scheduled without conflicts from the multicasts chosen earlier. Furthermore to reduce the impact of unicast traffic on the multicast performance of the Ωswitch, separate queues with priority schemes can be used at the outputs for multicast and unicast cells.

Bibliography

[1] Rajendra V. Boppana, C. S. Raghavendra, “Designing Efficient Benes and Banyan Based Input-Buffered ATM Switches,“ in *ICC*,1999.

[2] Nick McKeown et al., “The Tiny Tera: A Packet Switch Core,“ in *Hot Interconnects V, Stanford University*, August 1996.

[3] S. Yeh, M. G. Yluchyj, A. S. Acampora, “The Knockout switch: A Simple Modular Architecture for High-Performance Packet Switching,“ in *IEEE J. Select Areas Commun., SAC-5, No. 8, pp 1274-83*, Oct. 1987.

[4] Nick McKeown et al., “Matching Output Queueing with Combined Input and Output Queueing,“ in *Proceedings of the 35th Annual Allerton Conference on Communication, Control, and Computing. Monticello, Illinois*, Oct. 1997.

[5] Yijun Xiong, Lorne Mason, “Multicast ATM switches using Buffered MIN structure: A Performance Study,“ in *INFOCOM’97*.

[6] Nc McKeown, Balaji Prabhakar, “Scheduling Multicast Cells in an Input-Queued Switch,“ in *INFOCOM’96*.

[7] Matthew Andrews, Sanjeev Khanna, Krishnan Kumaran, “Integrated Scheduling of Unicast and Multicast Traffic in an Input-Queued Switch,“ in *INFOCOM’99*.

[8] M. J. Karol, M. G. Hluchyj, S. P. Morgan, “Input Versus Output Queueing on a Space-Division packet Switch,“ in *Proc. IEEE INFOCOM’92*.

[9] C. S. Raghavendra, R. V. Boppana, “On Self Routing in Benes and Shuffle/Exchange,“ in *IEEE Transaction in Computers*, Sept. 1991.

[10] M. J. Karol, K. Y. Eng, H. Obara, “Improving the performance of Input-Queued ATM packet switches“, *Proc. IEEE INFOCOM 1992*.

[11] T. Anderson et al., “High Speed Switch Scheduling for local area networks,“ in *5th Intl. Conf. Architectural Support for Programming Languages and Operating Syst*, Oct. 1992.

[12] Nc McKeown, P. Varaiya, J. Walrand, “Scheduling Cells in an Input-Queued Switch,“ in *Electron. Lett.*, Dec. 9, 1993.

[13] T. T. Lee, S. Y. Liew, “Parallel Routing Algorithms in benes-Clos Networks,“ in *Proc. INFOCOM*, 1996.

[14] Nc McKeown, “Scheduling algortihms for Input-Queued Cell Switches,“ *Ph.D. dissertation, Univ. California, Berkeley*, May 1995.

[15] M. Ali, H. Nguyen, “A neural network implementation of an input access scheme in a high-speed packet switch,“ *Proc. of GLOBECOM*, 1989.

[16] M. Chen, N. D. Georganas, “A fast Algorithm for Multi-channel/port traffic scheduling,“ in *Proc. IEEE Supercomm/ICC’94*.

[17] H. Obara, “An efficient contention resolution algorithm for input-queuing ATM switches,“ in *Int. J. Digital and Analog Cable Systems*, Oct.-Dec. 1989.

[18] A. Huang, S. Knauer, “Startlite: A wide band digital switch,“ in *Proc. IEEE GLOBECOM*, 1984.

[19] Mujumdhar S. P. et al., “Optimum Architecture for Input-Queuing ATM Switches,“ in *Electron. Lett.* March 28, 1991.

[20] Peter Newman, “ATM technology for corporate Networks,“ in *IEEE Communications Magazine*, April 1992.

[21] S. Q. Li, “Performance of a nonblocking space-division packet switch with correlated traffic,“ in *IEEE Trans. Commun.* Jan. 1992.

[22] Vipin Kumar et al., “Introduction to Parallel Computing, Design and Analysis of Algorithms,“ 1994.

[23] Nc McKeown, “iSLIP: A Scheduling Algorithm for Input-Queued Switches,“ in *IEEE Transactions on Networking*, April 1999.

[24] Mustafa K. Mehmet Ali, Shaying Yang, “Performance Analysis of a Random Packet Selection Policy for Multicast Switching,“ in *IEEE Transactions on Communications*, March 1996.

[25] Hyong S. Kim, “Multinet Switch:Multistage ATM Switch Architecture with Partially shared buffers,“ in *INFOCOM’93*.

[26] Gard and J. Rooth, “An ATM switch implementation technique and technology,“ in *Proc. Intl. Switching Symp.*, May 1990.

[27] A. Itoh et al., “Practical Implementation and Packaging technologies for a large scale ATM switching system,“ in *IEEE J. Select. Areas of Commun.*, Oct. 1991.

[28] F. A. Tobagi et al., “Architecture, Performance and Implementation of the Tandem Banyan Fast Packet Switch ,“ in *IEEE J. Select. Areas of Commun.*, Oct. 1991.

[29] F. A. Tobagi, “Fast packet switch for Broadband Integrated Services Digital Networks ,“ in *Proc. IEEE*, Jan. 1990.

[30] P. C.Wong, M. S. Yeung, “Design of a Novel Fast Banyan Switch - Pipeline Banyan,“ in *IEEE/ACM Transactions on Networking, Vol.3 No.1*, Feb. 1995.

[31] V. P. Kumar et al., “Phoenix: A building block for Fault-tolerant Broadband Packet Switches,“ in *Proc. IEEE GLOBECOM’91*, Dec. 1991.

[32] W. Wang, F. A. Tobagi et al., “The Christmas Tree Switch: An Output Queuing Space-Division Fast Packet Switch,“ in *Proc. of IEEE Infocom*, April 1991.

[33] A.Pattavina, “A Multiservice High-Performance packet switch for Broadband networks ,“ in *IEEE Tran. Commun., Vol. 38, No. 9 pp. 1607-15, ,* Sept. 1990.

[34] R. J.Proctor, T. S. Maddern, “Synchronous ATM switching Fabrics,“ in *Proc. International Switching Symp., Vol. 4, pp.109-14*, May 1990.

[35] Tamir. Y., G. Frazier, “High Performance multi-queue buffers for VLSI Communication Switches,“ in *Proc. of 15th Ann. Symp. on Computer Architecture*, June 1988.

[36] Adisak Mekkittikul, Nick McKeown, “A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches,“ in *IEEE Infocom 98, Vol 2, pp. 792-799*, April 1998.

[37] Adisak Mekkittikul, Nick McKeown, “A Starvation-free Algorithm for Achieving 100% Throughput in an Input-Queued Switch,“ in *ICCCN ’96, pp. 226-231*, October 1996.

[38] Nick McKeown et al., “Tetris Models for Multicast Switches,“ in *Proceedings of the Princeton Conference, Princeton*, March 1996.

[39] “CSIM user manual: http://ringer.cs.utsa.edu/faculty/boppana/courses/csim18man.htm“.

[40] Chao H. J, Choe B-S, “A large-scale multicast output buffered ATM switch,“ in *GLOBECOM*, 1993.

Vita

Ramakanth Gunuganti was born in Andhra Pradesh, India on August 30, 1977, the son of Bhaskar rao Gunuganti and Bharathi Gunuganti. He attended Little Flower Junior College in Andhra Pradesh, India, graduating in 1994. In 1998, he graduated from Regional Engineering College, Warangal, Kakatiya University, receiving the Bachelor of Technology with a major in Computer Science and Engineering. He entered the graduate program at UTSA in 1998.