INDIRECT INTERCONNECTION NETWORKS FOR

HIGH PERFORMANCE ROUTERS/SWITCHES

By

RONGSEN HE

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

August 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of RONGSEN HE find it satisfactory and recommend that it be accepted.

Chair

ACKNOWLEDGMENT

The achievement of this dissertation has been performed in conjunction with the High Performance Computer Systems (HiPerCopS) research group under the direction of Dr. José Delgado-Frias.

First of all, I would like to thank my advisor Dr. José Delgado-Frias for his guidance throughout my graduate study. His broad knowledge, enthusiasm and patience deeply impressed me. I would also like to extend my sincere appreciation to my other committee members Dr. Jabulani Nyathi and Dr. Sirisha Medidi. Their valuable advice and help greatly improved my research in the past years. Moreover, the author gratefully acknowledges the financial support as teaching assistant and research assistant received from the School of EECS at Washington State University. Especially, my thanks go to my advisor Dr. José Delgado-Frias, chair of graduate studies Dr. Ali Saberi and graduate secretary Ms. Ruby Young. I would also like to thank the WSU Graduate School for awarding me travel grants to attend the IEEE GLOBECOM 2006, which is one of the most prominent conferences in the networking and communication field.

I would like to convey my special gratitude to Professor K.C.Wang and Professor Nian-Feng Tzeng. Professor K.C.Wang helped me solve some programming problems in my research with his solid knowledge and extensive experience. Professor Nian-Feng Tzeng from University of Louisiana gave us a copy of his simulator source code, which helped our research at WSU.

Finally, I am indebted to my wonderful wife Lei Dong (董蕾) and lovely daughters Nancy and Lucy. Thanks for their endless love and unconditional support!

INDIRECT INTERCONNECTION NETWORKS FOR HIGH PERFORMANCE ROUTERS/SWITCHES

Abstract

by Rongsen He, Ph.D.
Washington State University
July 2007

Chair: José Delgado-Frias

Routers form the backbone of the Internet; their kernel, structure, and configuration (scheduler) of the backplane (or switching fabrics) dominate the routers' performance, scalability, reliability and cost. As higher performance is required with the rapid development of the network applications, router's architecture has also evolved from the shared backplane to switched backplane, which mainly uses the indirect interconnection networks.

The indirect interconnection networks include crossbar, MIN (multistage interconnection networks) and some other irregular topologies. At present, most of today's routers and switches are implemented on single crossbar with symmetric buffer architecture. In the first part of this dissertation, we introduce novel asymmetric buffer architecture for the crossbar in which a new port and a local shared bus are added. We then evaluate its performance and simulate under different bus arbitration and buffer management algorithms. Our studies indicate that we can get great improvement for the throughput and low drop rate. Thus we could save a lot of expensive link bandwidth and decrease the probability of congestion for the network.

Single crossbar complexity increases at $O(N^2)$ in terms of crosspoint number, which become unacceptable for scalability as the port number (N) increases. A delta class self-routing MIN with complexity of O($N \times \log_2 N$) has been widely used in the ATM switches. But the reduction of crosspoint number results in considerable internal blocking. A number of scalable methods have been proposed to solve this problem. One of them uses more stages with recirculation architecture to reroute the deflected packets, which greatly increase the latency. In the second part of this dissertation, we propose an interleaved multistage switching fabrics architecture and assess its throughput with an analytical model and simulations. We compare this novel scheme with some previous parallel architectures and show its benefits. From extensive simulations under different traffic patterns and fault models, our interleaved architecture achieves better performance than its counterpart of single panel fabric. Our interleaved scheme achieves speedups (over the single panel fabric) of 3.4 and 2.25 under uniform and hot-spot traffic patterns, respectively at maximum load (p=1). Moreover, the interleaved fabrics show great tolerance against internal hardware failures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

This dissertation is dedicated to my parents, my old brother, my wife and my lovely daughters Nancy and Lucy.

Chapter 1

Introduction

Routers form the backbone of the Internet. Their kernel, structure and configuration (scheduler) of the backplane (or switching fabrics) dominates the routers' performance, scalability, reliability and cost. As performance requirements increase as a consequence of the rapid development of the network applications, router's architecture has also evolved from the shared backplane to switched backplane [1], which mainly uses the indirect interconnection networks.

The interconnection networks provide communications between different nodes. The nodes could be processors, memory modules, Input/Output ports, telephones, computers, etc. According to the topology, the interconnection networks can be classified into direct interconnection networks and indirect interconnection networks, which are used for different applications.

1.1 Direct interconnection networks

The direct interconnection networks are also called static networks because the nodes communicate directly with each other through point-to-point links. Common examples of the direct interconnection networks are mesh, k-ary n-cubes and hypercube as shown in Figures 1.1, 1.2 and 1.3 respectively.

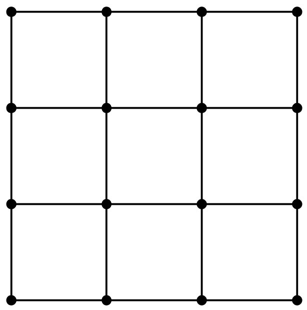
Figure 1.1: 2-D Mesh network.

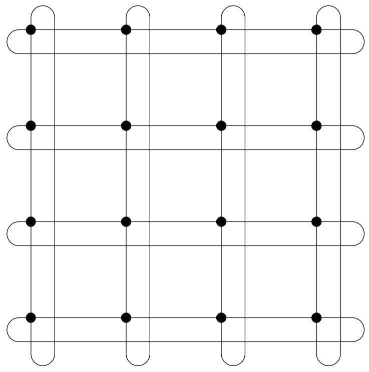
Figure 1.2: 4-ary 2-cube network.

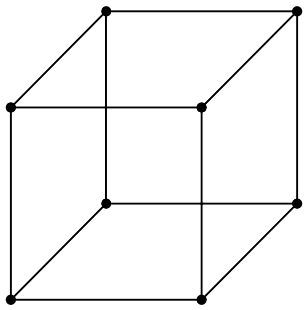
Figure 1.3: 3-D Hypercube network.

Because direct interconnection networks exploit the communication locality among nodes, they are widely used for System on Chip (SOC), high performance parallel and distributed systems (Multiprocessors or Multicomputers). For examples, the Intel Paragon computer systems adopt the 2-D Mesh to connect up to 2048 processors and the nCUBE parallel computers are based on the hypercube networks [2, 3].

1.2 Indirect interconnection networks

The indirect interconnection networks are also called dynamic networks because the nodes communicate with each other through some form of switching network. The connections between the nodes may be configured dynamically according to demands. Common examples of the indirect interconnection networks include shared bus, crossbar, multistage interconnection networks (MIN) and some other irregular topologies [4].

1.2.1 Shared bus

Buses are the most common type of interconnection networks because of its cheap hardware cost. Although it just allows only one pair of attached devices to communicate at any time, the performance is still adequate for many applications in computer system, such as ISA or PCI bus.

Thus, most routers until early 1990s are bus based for low cost. For example, Cisco 7000 series router was built on the Cbus (Cisco bus), which was operating at 16.67MHz and designed with 32 data lines, 24 address lines and 8 command lines. A simple diagram of the bus based router is shown in Figure 1.4. The routing scheduler was initially centralized; every packet must be processed by the central CPU. For some hardware improvement afterwards, the forwarding table is then distributed to each local linecard such that the forwarding decision is made locally and the packet could be transferred to the outgoing port immediately. However, the bus is shared between all ports, only one packet could be transferred between any two-port pair each time. This deadly bottleneck makes the bus superseded quickly by switched backplane such as crossbar in next section.

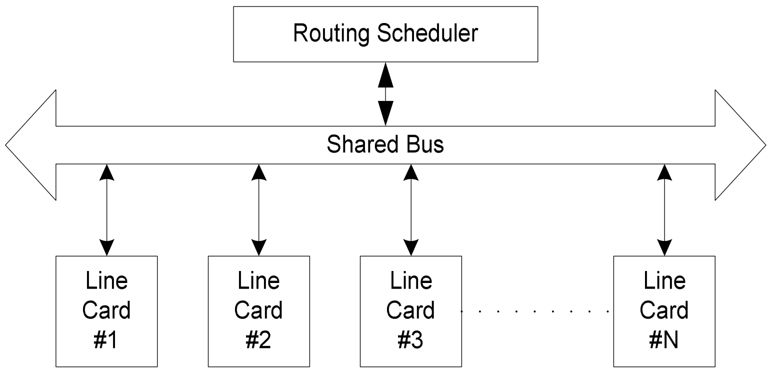

Figure 1.4: Shared bus network

1.2.2 Crossbar network

Crossbar network allows internally nonblocking connection from any idle input to any idle output. It is also the building block for multistage interconnection networks in the next section.

In Figure 1.5, the widely used crossbar structure for switched backplane in routers/switches is shown for simplicity. The connections between any two linecard become the configuration of the point-to-point links, which could be operated at very high speed. Moreover, packets could be transferred simultaneously between multiple port pairs. That is why most high performance routers today are implemented on this architecture, since it greatly increases the overall throughput of the network.

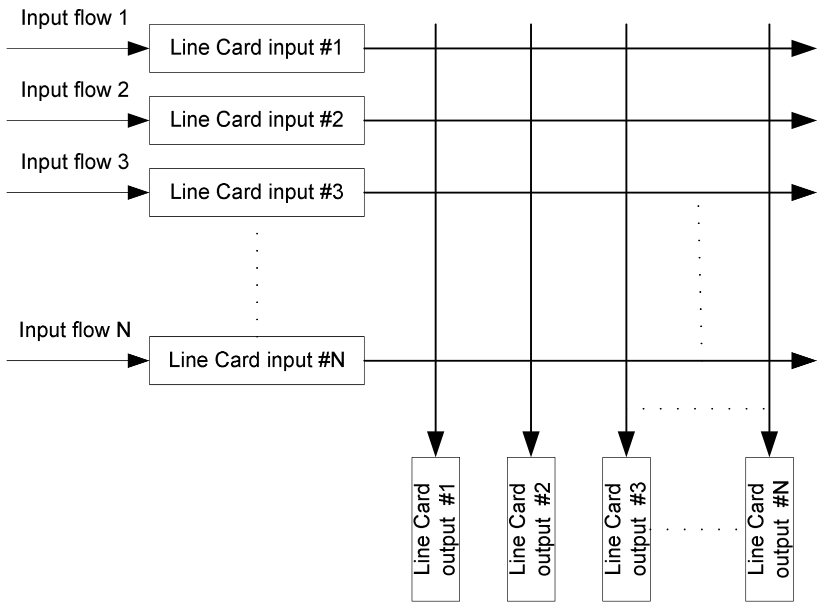

Figure 1.5: Crossbar network.

1.2.3 *Multistage interconnection networks*

As stated above, the crossbar outperforms the shared bus. However, a crossbar requires O(N^2) crosspoints to interconnect N ports, which is very expensive for the hardware cost. This prompted many researchers to explore new methods to reduce the crosspoint number and introduced the multistage interconnection networks in the past decades [5, 6]. According to their capability, they can be classified into nonblocking multistage networks and blocking multistage networks.

1.2.3.1 Nonblocking multistage networks

The basic three-stage Clos network [5] brought in 1953 is built with smaller crossbar switches. It is depicted in Figure 1.6.

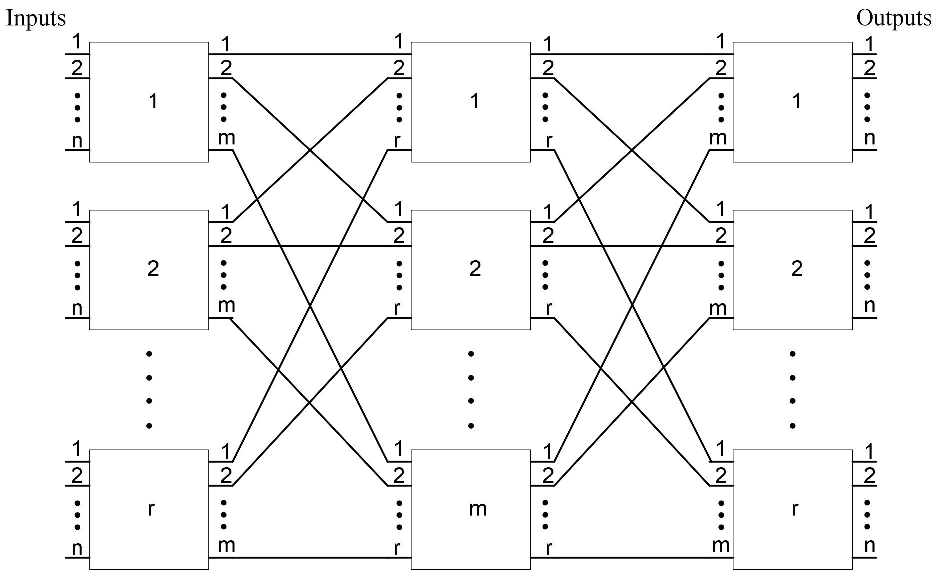

Figure 1.6: A general three-stage Clos network.

From Figure 1.6, the Clos network demonstrates different interconnection capability depending on the parameters ***n*** and ***m*** [5, 6].

1. Strictly nonblocking networks

A network is called strictly nonblocking network if it can connect any idle input to any idle output regardless of what other connections are currently in progress.

Obviously, the crossbar is a strictly nonblocking network. As to the three stage Clos network, when $m \geqslant 2n-1$, it is a strictly nonblocking network.

2. Wide-sense nonblocking networks

A network is called wide-sense nonblocking network if it can connect any idle input to any idle output without changing other current connections, but it can do so only if it must follow specific routing rules.

As to the three stage Clos network, when $\boldsymbol{m \geqslant (3/2)n}$, it is a wide-sense nonblocking network. Moreover, the wide-sense nonblocking network has fewer crosspoints than the strictly nonblocking network with the same total port number N.

3. Rearrangeable nonblocking networks

A network is called rearrangeable nonblocking network if it can connect any idle input to any idle output by rearranging some current connections so that a new connection can always be established.

As to the three stage Clos network, when $\boldsymbol{m \geqslant n}$, it is a wide-sense nonblocking network. Due to the flexibility and relatively low cost, the rearrangeable nonblocking networks have been widely studied sine 1950s. However, the routing complexity restricts its growth when port number N becomes large. I will discuss about it in Chapter 6.

1.2.3.2 Blocking multistage networks

A network is called blocking multistage network if it can connect many, but not all, possible connections between idle input-output pairs.

Typical blocking multistage networks include banyan, omega, baseline and reverse baseline, and indirect binary n-Cube networks. Examples of 8×8 baseline, reverse baseline, and indirect binary n-Cube networks are depicted in Figures 1.7 to 1.9 respectively.

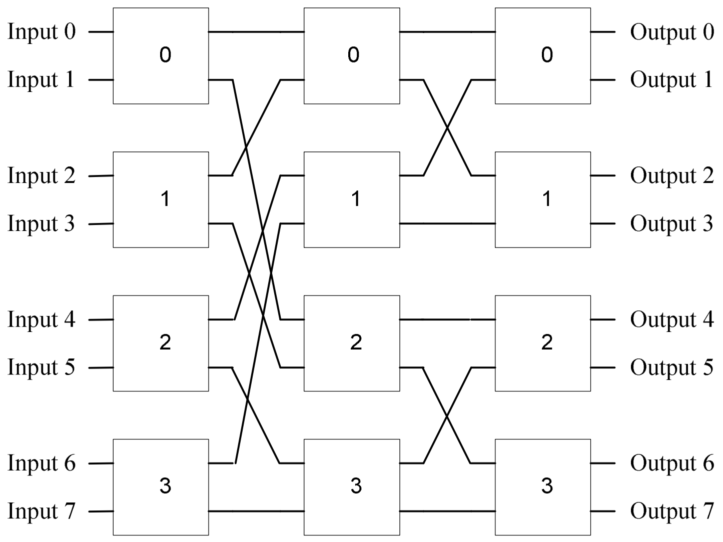

Figure 1.7: 8×8 baseline network.

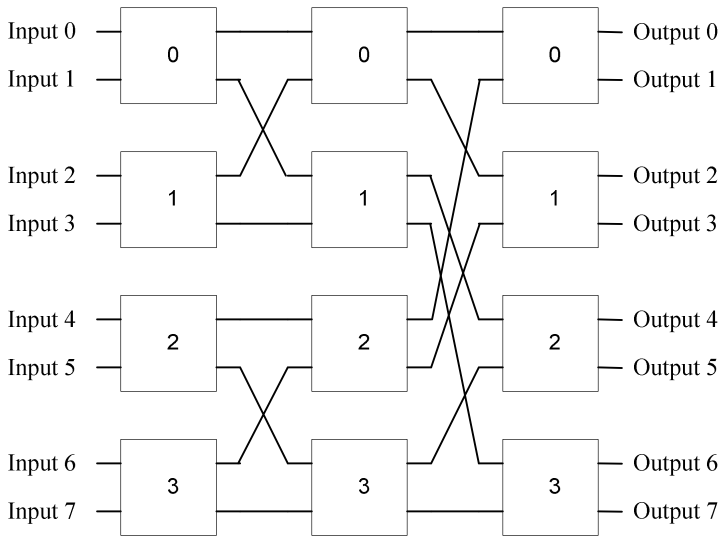

Figure 1.8: 8×8 reverse baseline network.

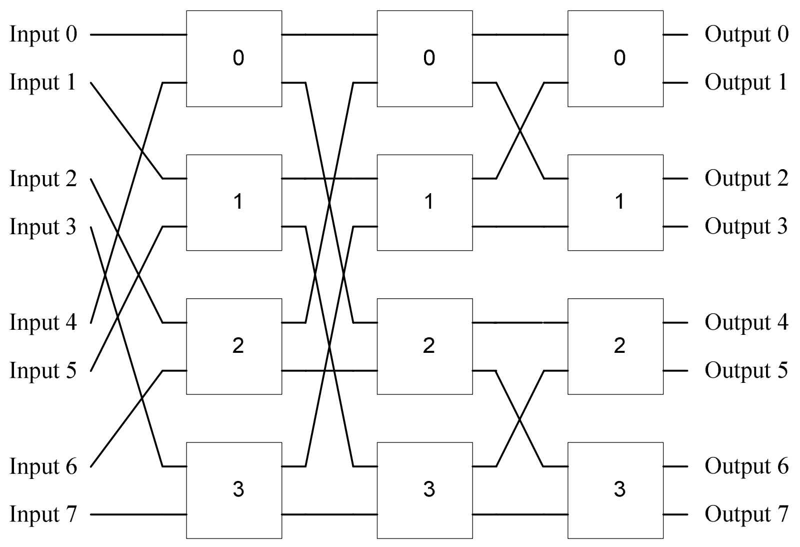

Figure 1.9: 8×8 indirect binary n-Cube network.

Banyan, omega, baseline and reverse baseline, and indirect binary n-Cube networks have different connections between each stages, even they have $\log_2 N$ stages consisting of 2×2 switching element. However, they are all topologically equivalent [7, 27]. Tow multistage interconnection networks are said to be topologically equivalent means that we could rearrange the positions of the switching elements of one network to get another network identical in topology except the input and output terminal numbers. The equivalence is very important because the properties of one network are always applicable to others such as the self-routing from inputs to outputs.

Though the blocking multistage networks can not implement all the permutations between inputs and outputs, the simplicity of self-routing with smaller number of switching elements make them very attractive in router and switch design. I will discuss about it in Chapter 3 to 5.

1.3 Outline

Because indirect interconnection networks can provide a variety of global communication paths among nodes, they are widely used for parallel computers, switching and routing systems. Thus in our research, we will address their applications in high performance routers/switches including:

- Overflow buffer architecture in crossbar.
- Fault tolerant switching fabrics with interleaved multistage interconnection networks.

Here, my researches focus on crossbar and multistage interconnection networks separately because they are feasible for different application environments of switching and routing. The remainder of this dissertation is organized as follows.

Chapter 2 describes the details of overflow buffer architecture in crossbar. We will evaluate its performance with extensive simulations for different bus arbitration and buffer management algorithms. Our studies indicate that this novel architecture provides higher throughput and maximizes the utilization of the expensive long-haul links.

In Chapter 3, I propose a novel scalable architecture of interleaved multistage switching fabrics. A theoretical model has been developed to analyze throughput of the fabrics. We also compare it with previous parallel architectures and show its benefits.

To evaluate performance of the interleaved architecture, extensive simulation results and analysis are presented in Chapter 4. They also match with the theoretical model given in Chapter 3

We also demonstrate our scheme's great fault tolerance under different fault models in Chapter 5. Based on these properties, we bring out the new concept RAIF (Redundant Array of Independent Fabrics), which could get better performance and fault tolerance as RAID system [8].

Finally, I present the conclusions in Chapter 6 and summarize the contributions of this research for next generation networks. The appendix lists the papers published during the course of my research.

Chapter 2

Overflow buffer architecture in crossbar

2.1 Research background

The initial backplane basically use a shared bus or memory. However, because the bus or memory is shared between all ports, only one packet could be transferred between any two-port pair each time. In order to overcome this bottleneck, switched backplane has replaced its predecessor. In switched backplane, the crossbar structure is widely used for its simplicity. The connections between any two linecards (LCs) become the configuration of point-to-point links, which can be operated at very high speed. Moreover, the crossbar is internally non-blocking because packets can be transferred simultaneously between multiple port pairs. This architecture greatly increases the overall throughput of the network and decreases the delay at the same time.

Another major issue that needs to be considered is the buffer bandwidth. Because the bursty characteristic of Internet flows, buffers in router are required to accommodate the packets or cells temporarily. Normally, the shared backplane uses a large shared memory as Juniper M40 router. A router, with N linecards each connected to a line at rate R, needs a bandwidth of 2NR. On the other hand, the non-blocking crossbar switch has a distributed buffer; each linecard has a buffer either at input or output. Although the output buffering could achieve better throughput, it requires the N×N switch to run N times faster. Consequently, most high performance routers use input queues with the crossbar [9]. Under this input queuing, each memory needs only run at a rate 2R (instead 2NR). Crossbar switch outperforms the shared backplane given the same memory speed.

The router buffers are usually built using cheap DRAM, which is optimized for large size rather than fast speed. Over the last decade, the speed (access time) of the commercial DRAM has increased only 1.1 times every 18 months, while the capacity of commercial routers has increased by about 2.2 times every 18 months. Thus, buffer architecture and management has become one of the most challenging issues of the design for the next generation routers, even using the crossbar switch.

With today's DRAM technology, it is barely possible to design buffers for a 40 Gb/s linecard. As the link rate increases, researchers give different solutions for this problem. A solution is to use a hierarchical memory [10, 11] where a small and fast SRAM is used as cache and a large and slow DRAM as main memory for buffer. The SRAM holds the heads and tails of the packet FIFOs, so the buffer could meet the increasing link rate to receive and send packets. With this two level buffer hierarchy, a 100TB/s packet-switched router is being designed [12], based on Chang's load-balanced switch architecture [13]. Though the load-balanced switch guarantees 100% throughput for a broad class of traffic, it uses two full crossbars as switching fabric, which increases the complexity, power and hardware cost a lot. So we are still not sure about the practicability of this architecture in the future network market. For single stage switch architecture, a model of Distributed Shared Memory router is introduced in [14]. The buffers on a linecard do not necessarily hold packets that arrived from, or are destined to that linecard. The buffers are shared and distributed. So this architecture needs lower memory bandwidth and fewer memory modules. Though the authors believe it to be a promising architecture, it needs higher crossbar bandwidth (significant crossbar speedup is either uneconomical or impossible). And it is difficult for the scheduler to find the available memory for each packet in turn.

2.2 Proposed overflow buffer architecture in crossbar

Based on these restrictions above, we propose a new overflow buffer architecture, which is shown in Figure 2.1. Comparing with the shared backplane and switched backplane, we find that the standard crossbar is N×N symmetric while the crossbar in Figure 2.1 is (N+1)×N asymmetric (N+1 inputs and N outputs) after adding the overflow buffer. The overflow buffer also occupies one input port connected with the crossbar. We use a multiplexer at the ingress of each linecard. When the buffers at that linecard have space, the incoming packet will be stored in FIFO (First In and First Out) queues of that linecard. Only when the buffers of that linecard are full and the packet is blocked, the multiplexer will switch the flow to request the shared bus. Then the shared bus arbitrator will select one from all requesters as the winner during each turn, which finally could be transferred to FIFO queues in the overflow buffer. Other requesters will be dropped because of collision.

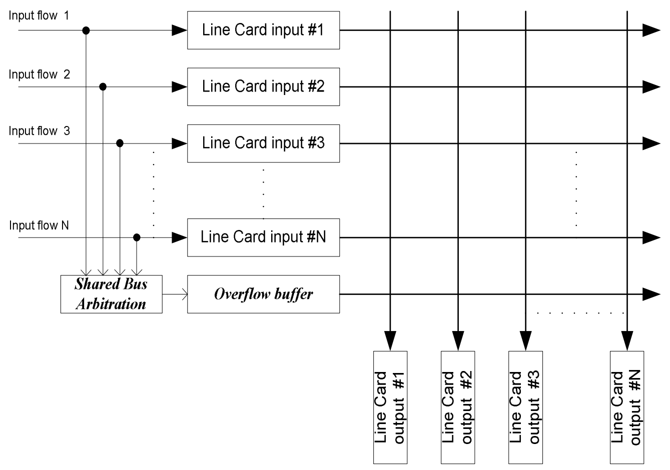

Figure 2.1: Overflow buffer backplane

Our objective here is to decrease the need for high crossbar bandwidth in [14], though we could not share all the buffers at all linecards. Adding a shared bus and the overflow buffer which is public for all the linecards, it is possible to improve the throughput since all linecards could statistically share one more path to the outputs when there is congestion. Obviously, the new overflow buffer architecture combines the merits of shared backplane and the switched backplane together. Because of the bursty characteristic of the Internet flows, the statistical sharing of the resource will result in performance improvement. And most important, the hardware cost is just incremented a little with O(N), compared with O(N^2) [12, 13]. In the following sections, we will evaluate this new architecture.

2.3 Operation of the overflow buffer architecture

Because our overflow buffer architecture has a crossbar and a shared bus, a scheduling algorithm is used to set the switch configuration for the crossbar, and bus arbitration for the shared bus. It is crucial to make an assumption throughout this chapter that the crossbar and the shared bus are of fixed length cell based, though the cell length here is not necessarily as that of ATM cells. The variable length packets must be segmented into the fix sized cells before being transferred across the backplane. So the scheduler could make fair decision among all connections and maintain efficient use of the crossbar and shared bus [1, 9]. The cells are reassembled back into previous packets at the output before being sent to the outgoing line.

Input queuing is subject to the HOL (Head of Line) problem with maximum throughput of 58.6% with FIFO queues [15]. This problem is eliminated by a solution, called VOQ (virtual output queuing) [16]. At each input, VOQ constructs separate FIFO queues for different outputs so that a cell will not be blocked by cells to other destination. In our (N+1)×N asymmetric crossbar structure, we will construct N VOQs at each of the N+1 input port (N normal input plus one overflow buffer). Thus, the buffer at each of N normal inputs will be divided equally among N VOQs corresponding to N outputs. As for the overflow buffer, space can be used by flows from all inputs. This requires more subtle buffer management to make sure that they are used efficiently between bursty flows and prevent abuse from some malicious flows. We will discuss it in the following part.

2.3.1 Scheduling algorithm for crossbar

The scheduling algorithm decides the time that a cell in the buffers of the N+1 inputs could be transferred to the destination outputs. Nowadays, a few scheduling algorithms have been developed for the input buffered crossbar switches using VOQs. Most of them attempt to achieve high throughput by looking for maximum bipartite matching between inputs and outputs. Schemes such as PIM [17], iSLIP [18, 19] repeatedly search for matches at each time of scheduling. This iterative scheduling algorithm consists of three major steps in each iteration:

- **Request stage:** An input buffer may have several requests. Each unmatched input sends requests to the outputs for which it has cells.
- **Grant stage:** There may be several requests to an output. Each unmatched output chooses one from several received requests and sends a grant signal to one of the inputs.
- **Accept stage:** Each unmatched input may receive grant signals from several outputs. Upon receiving grant signals, each input sends an accept signal to only one of the outputs offering the grants.

PIM is the first switch scheduling algorithm that employs an iterative approach, and was developed by DEC Systems Research Center for the 16-port, 16Gb/s AN2 switch. At the grant stage, each output selects randomly a requesting input. Again, at the accept stage, each input accepts one output from several grant signals at random. The random selection of the grant and accept stages may cause an input to be starved for a long time. Moreover, implementation of random selection among a time-varying set is very complicated.

iSLIP is used for the Cisco 12000 high performance routers. The iSLIP scheme uses rotating round-robin priority arbitration to schedule active inputs and outputs in turn. A grant pointer is kept for each output to track the input with the highest priority.

Similarly, there is an accept pointer for each input which tells the output with the highest priority. Whenever a match is found, the corresponding grant/accept pointers are incremented.

Compared with PIM, iSLIP is simpler in implementation and achieves higher throughput. Due to the desynchronization between arbiters, it also provides 100% throughput under traffic with uniform distribution for any number of iterations [20]. So we have chosen iSLIP as the scheduling algorithm for crossbar in our overflow buffer architecture.

2.3.2 Arbitration for the shared bus

When the incoming cells are blocked by input port FIFO queues, it means that the buffer space of that VOQ is full. In conventional schemes, the incoming cells are discarded until buffer space is available. In the proposed scheme, each input could access the overflow buffer in case of temporary congestion. The overflow buffer works as a reservoir to accommodate the heavy bursty flows. But as two or more congested flows from different inputs attempt to use the shared bus, bus arbitration is required to solve collisions. Though the shared bus works as a local area network, the method CSMA/CD used in IEEE 802.3 could not work here because we must transfer the cells according to the cell time. Here, we propose three arbitration methods.

1. Priority hierarchy

The input ports are arranged with a static priority hierarchy. For example, to a 16-port switch, port 1 has the highest priority and port 16 has the lowest priority. When collisions happen between the input ports, the arbitration will select the cell from the input port with the highest priority and drop others. The reason to use the priority hierarchy is to differentiate the service provided for different link users. At the same time, priority hierarchy is the easiest to be implemented in hardware. But the input port with the lowest priority may be starved.

2. Round-robin (RR)

If all the input ports have equal priority, round-robin arbitration is used to resolve conflicts. The round-robin rotating priority is similar to iSLIP's one. A selection pointer is kept for the shared bus to track the input port with the highest priority. Whenever a selection decision is made, the selection pointer is incremented by one location. So the lowest priority is always given to the most recently connected input port. Because of the round-robin moving of the selection pointer, the algorithm provides a fair allocation of bandwidth among all input ports.

3. Round-robin with global information (RRG)

Based on the round-robin algorithm, we could improve the performance by combining the information of occupancy in the buffer. If the buffer space assigned to a specific FIFO in the overflow buffer is full, we will drop the cell anyway even after it is transferred to the overflow buffer. Knowing the occupancy of each FIFO in the overflow buffers, the arbitration scheme will select the cell from the input port with the highest priority, which has free space in overflow buffer. This implementation needs some global information; in return it efficiently saves expensive bus bandwidth.

2.3.3 Overflow buffer management

The buffer management is the policy to allocate the buffer space. N VOQ FIFO queues are also maintained in the overflow buffer. The following three buffer management algorithms will allocate space for each FIFO queue.

1. All space private (PRIVATE):

In this algorithm, all the overflow buffer space is allocated equally among all FIFO queues, just as the allocation method in other input port. One flow could not use the free space of other flows. This provides the mechanism to protect benign flows from the malicious one.

2. All space public (PUBLIC):

This algorithm will thoroughly share the entire overflow buffer. Any cell is accepted as long as the overflow buffer has free space. Using cheap DRAMs, it is easier to add memory to the overflow buffer than to each input port without increasing the cost sharply.

3. Half space public and half space private (PUBLIC-PRIVATE):

Here, half of the overflow buffer space is allocated equally among all FIFO queues and another half is used for public. Any flow could use the public space when there is a shortage of its own. This more complex algorithm provides a protection mechanism against a flow using the entire buffer space.

2.4 Evaluation of the overflow buffer architecture

The performance of the overflow buffer architecture depends on flow mode, buffer size and the size ratio between the overflow buffer to buffer in each input. A cell-based simulator is used for our evaluation; this is based on a simulation tool called “SIM” [21]. The main metrics we used for evaluation are cell drop rate and average delay. They represent the most important performance of a switch/router. The cell drop rate is defined to be the ratio of the number of lost cells to total number of cells arriving at the switch from outside. The average delay is defined to be the average cell times that the cells stay at the switch.

Our simulation is based on a 16×16 switch (the overflow buffer is transparent to the outside world, so it is a 17×16 asymmetric crossbar in reality). There are 16 flows per input of crossbar each destined for a different output, for total 17×16=272 flows. The simulation time is set to 100,000 cell times; this time is large enough for the results to converge. Because the overflow architecture is just used under congestion, there is little difference between the standard symmetric 16×16 crossbar and the 17×16 asymmetric crossbar when the offered traffic load is light and smooth [20]. During the simulation, we should saturate both architectures with total 16 overloaded flows for input. Our simulation results have been grouped on two flows: The first group consists of 8 “bursty” flows and 8 “bernoulli_iid_uniform” flows, which represents the normal running condition of the network. The second group just consists of 16 “bursty” flows, which represents the worst condition of the network. All the flows are overloaded to 92%.

Detailed information on the simulator parameters can be found in the SIM manual [21]. The main parameters are listed below:

1. **Architecture:**
   - Standard 16×16 symmetric crossbar.
   - 17×16 asymmetric crossbar.
2. **Flow mode:**
   - Bernoulli-bursty: 8 bursty flows and 8 bernoulli flows.
   - Bursty: 16 bursty flows.
3. **Size ratio between overflow buffer and each input buffer:**
   - In 16×16 symmetric crossbar, obviously the ratio is 0.
   - In 17×16 asymmetric crossbar, the ratios are 1, 2, 4 to 8.

2.4.1 Comparison between the two architectures

In this section, we compare the performance when both architectures have the same hardware resources (i.e. total buffer size). The buffer allocation is based on ratio (defined as overflow_buffer_size/input_port_buffer_size). As the size ratio increases, more resource will go to the overflow buffer. The simulation results are shown in Figure 2.2 to Figure 2.4. Here, we use the priority hierarchy as the shared bus arbitration. The three curves correspond to three different total buffer sizes. The number in the figure's legend indicates the total buffer size.

From Figure 2.2(a) and (c), it can be observed that after adding an overflow buffer, the cell drop rate falls down about 2 percent under both flow modes when ratio is 1 or 2. This in turn means we have a better utilization of available bandwidth as well as less need of retransmission. We also could see that ratio 1 or 2 is optimal point for hardware design since performance degrades afterwards.

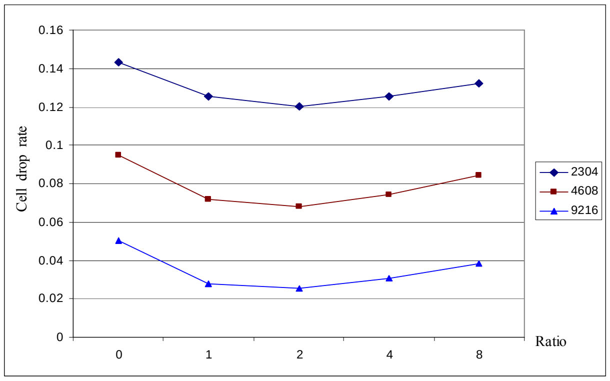

Figure 2.2(a): Architecture comparisons: Drop rate vs Ratio (Bernoulli-bursty mode).

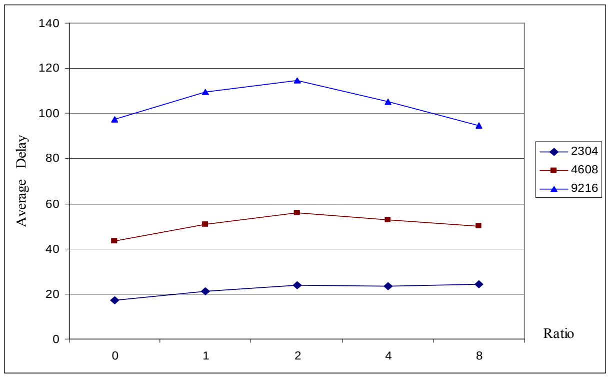

Figure 2.2(b): Architecture comparisons: Delay vs Ratio (Bernoulli-bursty mode).

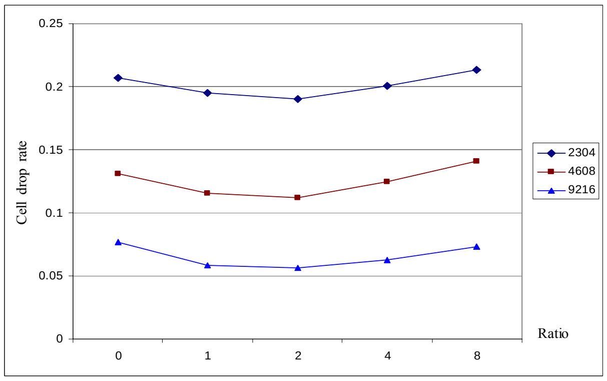

Figure 2.2(c): Architecture comparisons: Drop rate vs Ratio (Bursty mode).

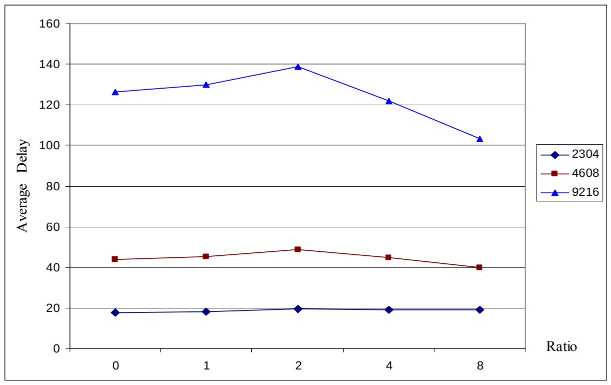

Figure 2.2(d): Architecture comparisons: Delay vs Ratio (Bursty mode).

On the other side, from Figure 2.2 (b) and (d), the average delay increases a little at the same time. Since the drop rate decreases, there are more cells that need to be transferred. The number of outputs is not increased; we have essentially the same bandwidth at the output. This makes the delay to increase slightly. According to the balance between the gains and lost, the overflow buffer architecture is preferred because it mediates the bursty flows under the worst congested conditions, and works as the standard architecture when the traffic is smooth.

2.4.2 *Finding the optimum size ratio*

Based on the previous results, it is shown that the overflow buffer with the same total buffer size has potential. In the next set of simulations (The priority hierarchy as the shared bus arbitration is still used), we will give the same FIFO size for both architectures. The number in the figures represents the FIFO size in each input port. And the overflow space is divided equally among all FIFO queues. The shared overflow buffer has a larger memory than an input port (this is given by the ratio). Considering that the DRAM's price still continues falling down, we could integrate much more overflow buffer with the crossbar without affecting the scalability of the linecards.

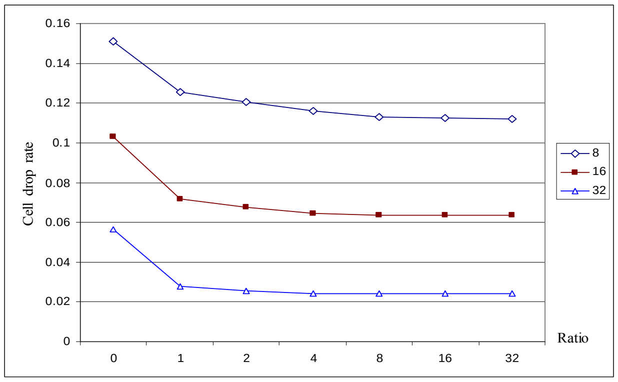

Figure 2.3(a): Optimum ratio: Drop rate vs Ratio (Bernoulli-bursty mode)

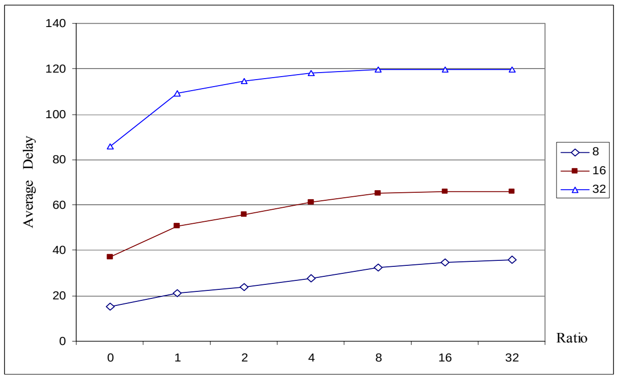

Figure 2.3(b): Optimum ratio: Delay vs Ratio (Bernoulli-bursty mode)

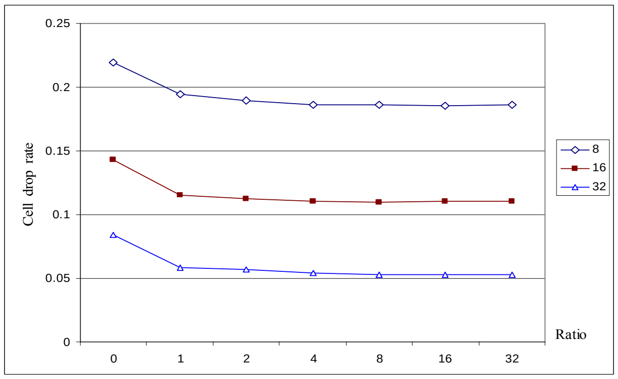

Figure 2.3(c): Optimum ratio: Drop rate vs Ratio (Bursty mode)

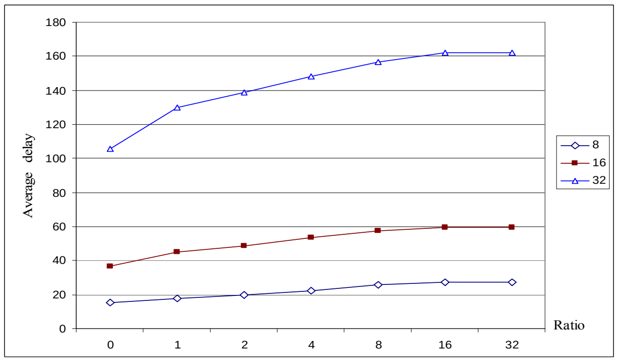

Figure 2.3(d): Optimum ratio: Delay vs Ratio (Bursty mode)

As observed in Figure 2.3, when the ratio increases, more hardware resource will be added to the overflow buffer. But we could see that after the overflow buffer becomes large enough, the simulation results will converge. This means infinite overflow buffer will not help improve the performance, just waste hardware memory. Considering that the marginal gain in performance is dropping, ratio 1 or 2 is preferred for real design though they are not the converging point. Moreover, the performance improvement at ratio 1 is gained by our new architecture. The following is gained by increasing the hardware cost.

2.4.3 The shared bus arbitration

In chapter 2.3.2, we proposed three shared bus arbitration: priority hierarchy (priority), round-robin (RR) and round-robin with global information (RRG). Each one is more complex than its predecessor. Because of the complexity for implementation, we do not use a random arbitration (as in PIM) here because it needs random selection among a time-varying set. The simulation results of the three arbitrations are shown in Figure 2.4. The number in the figures represents the FIFO size in each input port. The overflow space is divided equally among all FIFO queues.

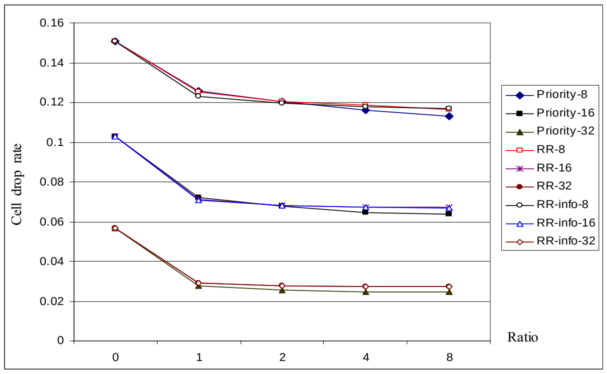

Figure 2.4(a): Shared bus arbitration: Drop rate vs Ratio (Bernoulli-bursty mode)

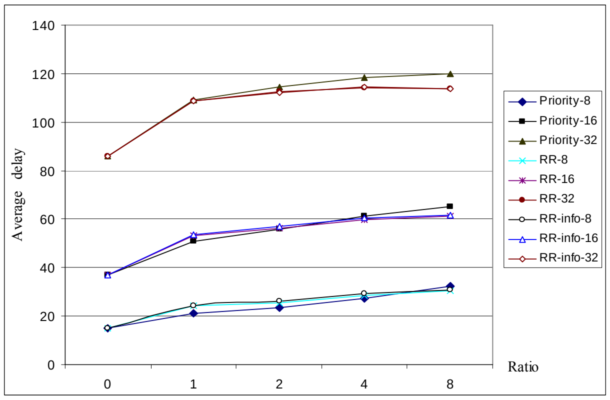

Figure 2.4(b): Shared bus arbitration: Delay vs Ratio (Bernoulli-bursty mode)

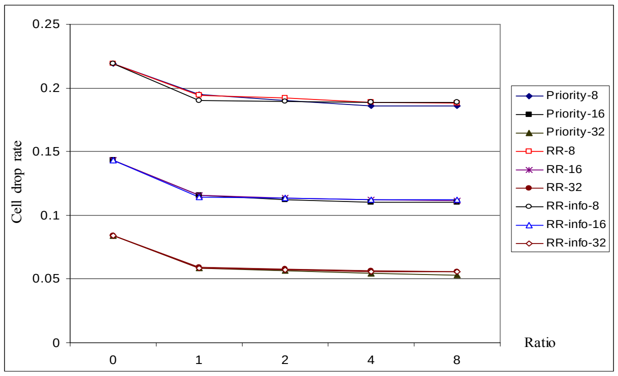

Figure 2.4(c): Shared bus arbitration: Drop rate vs Ratio (Bursty mode)

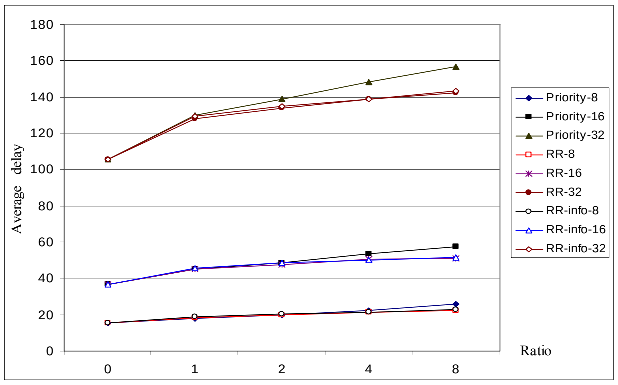

Figure 2.4(d): Shared bus arbitration: Delay vs Ratio (Bursty mode)

From Figure 2.4, the priority hierarchy could get the same cell drop rate under bursty flow mode and even better under the bernoulli-bursty flow mode. On the other hand, average delay becomes worse in particular under the bursty flow mode. Because the bursty flows with higher priority will always keep the shared bus and congest its own FIFO in the overflow buffer. With the RR arbitration, it will balance the bursty flows to efficiently make use of the FIFO queues in the overflow buffer, helping the iSLIP get more maximum match and reduce the average delay.

The difference between the RR and RRG is not significant from the simulation. RRG gets slightly better results only when the FIFO size of the overflow buffer is small.

When FIFO size of the overflow buffer is large enough, the results converge together since every incoming cell to the overflow buffer could always find its space. On the other hand, RR tries to scatter the flows into the FIFO queues of overflow buffer. So the probability that one FIFO is blocked will drop quickly with a little increase of the FIFO size. Nowadays, considering a 40 GB/s switch port with 40-byte cells and a speedup of two, the arbitrator has only just 4 ns to resolve the contention [22]. So, we should try to make the arbitrator as simple as possible. Although RRG is better, RR may be preferred in an actual implementation because of its simplicity. RR hardware implementation can be combined with the iSLIP's scheduling arbiters, which makes the design easier.

2.4.4 The overflow buffer management

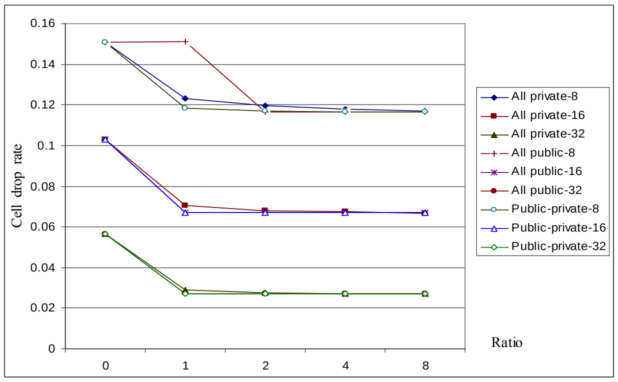

Figure 2.5(a): Overflow buffer management: Drop rate vs Ratio (Bernoulli-bursty mode)

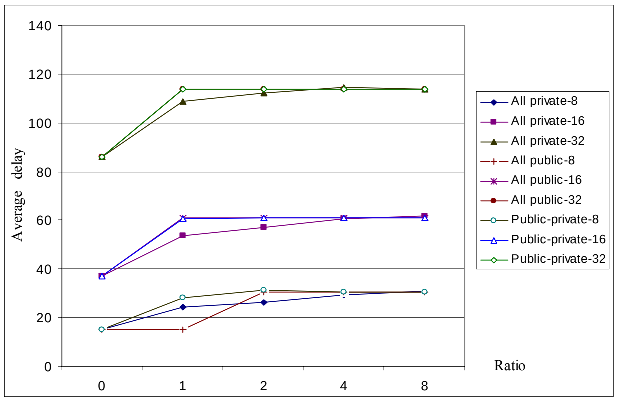

Figure 2.5(b): Overflow buffer management: Delay vs Ratio (Bernoulli-bursty mode)

In Section 2.3.3, we proposed three buffer management algorithms which will allocate the space for each FIFO in the overflow buffer. In the simulation above, we just equally distribute the overflow buffer and make the space private to each FIFO. In this part of simulation, we will compare it with other two buffer management algorithms. The shared bus arbitration used in the simulation is RRG. The results are shown in Figure 2.5.

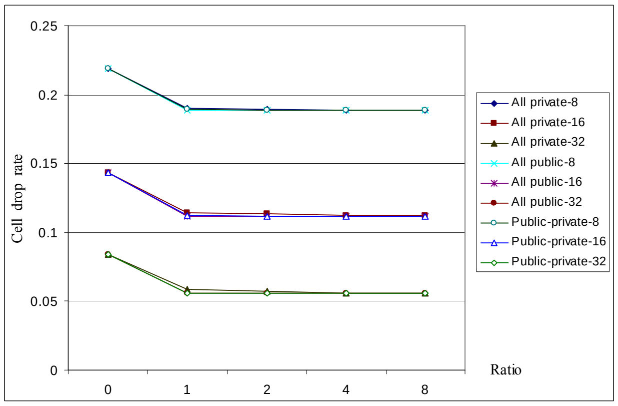

Figure 2.5(c): Overflow buffer management: Drop rate vs Ratio (Bursty mode)

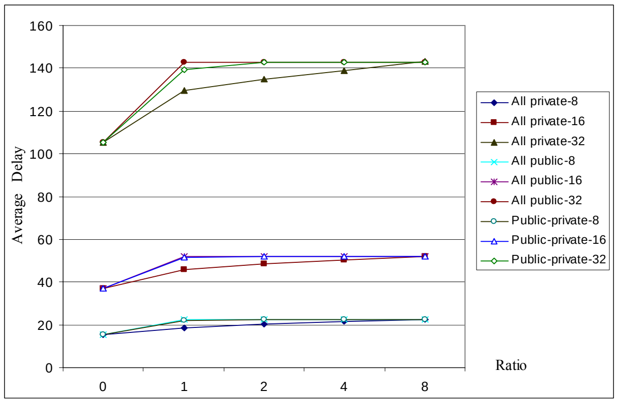

Figure 2.5(d): Overflow buffer management: Delay vs Ratio (Bursty mode)

From Figure 2.5, it can be observed that performance converges when the overflow buffer becomes large enough. On the other hand, PUBLIC method degrades performance a lot under small overflow buffers, especially for the bernoulli-bursty flow mode in which there is a singular point. Because this method could not provide the mechanism to protect flows from the malicious ones. The PUBLIC-PRIVATE gives a little lower cell drop rate than the PRIVATE method, but more delay latency. Because the congested flows could use more efficient space in the public area without being dropped and they need more time for scheduling.

Summing up, PUBLIC method can use the hardware more efficiently; however, a designer may still need to use the PUBLIC-PRIVATE or PRIVATE methods to prevent some malicious flows from disturbance, in particular when there is small overflow buffer. The allocation that half space public and half space private here is just ease for simulation. The designer could modulate the percentage between the private and public space according to their requirement.

2.5 Summary

In this part, we have explored new asymmetric buffer architecture for the crossbar in which a new port and a local shared bus are added. We have shown that this new overflow buffer architecture could get much lower cell drop rate; latency is increased a bit at the same time. The proposed scheme could save a lot of expensive link bandwidth by avoiding retransmission in TCP/IP. Size ratios of 1 or 2 may be preferred according to the marginal gain to hardware cost. We also examined three shared bus arbitrations. Our simulations show that RR arbitration offers both simplicity and high performance. Three buffer management methods are also studied here. Buffer allocation based on private reservation for FIFO queues offers better performance and mitigates the influence of malicious flows.

Longer delay latencies are obtained with the overflow buffer architecture in the simulations. This is due to significant reduction on drop rate. In order to provide the QoS for some special flows, we could incorporate multiple classes of traffic with different priority levels into this architecture. A prioritized iSLIP scheduling algorithm is also given in [18]. It is worth mentioning that we could combine them together to meet the QoS requirement. So we believe that our overflow buffer architecture will also meet the diversified services on the Internet for the future applications.

Chapter 3

Scalable interleaved switching fabrics

3.1 Research background

The advantage of packet switching with statistical multiplexing makes the convergence inevitable of the Internet, telecommunication and TV service. For example, in the past few years, Britain has updated its entire telephone network to the Internet Protocol [23]. So there is no technical difference between the telephone network and the Internet in UK. At the same time, both telecommunication carriers and cable companies provide integrated voice, video, and data service for their customers with IPTV [24], which uses IP network to deliver TV program. Thus, the incorporation of next generation network requires that a large number of line cards be integrated in a single high performance router. However, most of present routers are based on single stage crossbar, which suffers from the scalable complexity with $O(N^2)$ (N is the fabric size or input/output number). As a result, these routers only support up to 16×16 interconnection in real applications such as the Cisco 12000 high-end router.

Multistage interconnection networks, such as Banyan [25], Omega [26], Baseline and reverse Baseline [27], and indirect binary n-Cube [28], belong to the delta class network that was defined by Patel [29]. They were firstly proposed for large multiprocessor system, which was the hardware foundation of the supercomputers. The purpose is to interconnect processor-to-processors and processor-to-memory modules for fast parallel computation. The delta class network has two special properties: unique path between each input-output pair and self-routing in each intermediate stage. Because of the simplicity of its self-routing without a complex scheduler, the delta class network is very attractive for the design of high speed switching fabrics. However, the reduced complexity to $O(N\times\log_2 N)$ also comes with the expense of serious internal blocking, which leads to a poor throughput under some traffic patterns. The basic reason for this drawback is that the delta class is not a permutation network; it could not implement all the permutations of inputs with a single copy ($\log_2 N$ stages) of such a network. In [30], Wu and Feng concluded that $3\times(\log_2 N)-1$ stages through the regular shuffle exchange network are sufficient to realize arbitrary permutation. In [31], Feng and Seo reduce this limit to $2\times(\log_2 N)$-stage shuffle exchange network (or two copies of Omega networks). Unfortunately, their inside-out routing is found to be incomplete [32]. In recent years, Çam [33] proves the rearrangeability of the asymmetric $(2\times(\log_2 N)-1)$-stage shuffle exchange network, which has the same permutation capability as the symmetric Beneš network [6, 34]. However, no simple routing algorithm has been developed for the asymmetric $(2\times(\log_2 N)-1)$-stage shuffle exchange network. And for the symmetric Beneš network, a number of routing algorithms [34-39] have been developed to realize arbitrary permutations between inputs and outputs in the past decades. These complicated algorithms required a number of computations that made it unfeasible for a simple and fast hardware implementation and not scalable for future generation of network routers.

An output-queued MIN with b×2b switching elements was studied in [40, 41]. The number of cells that can be concurrently switched from the inlets to each output queue equals to the number of stages in the interconnection network. Tzeng [42] improves the architecture [41] by choosing different recirculation approaches from the last copy of stages. But in order to achieve higher throughput, more stages are required, which increase both the latency and hardware cost.

In this chapter, we propose a novel scheme that uses interleaved multistage switching fabrics. This architecture keeps the Switching Element (SE) hardware complexity acceptably low while achieving better performance under various traffic patterns. Moreover, the architecture provides great tolerance against internal hardware failures which is beneficial in the critical environment of network infrastructure.

The rest of this paper is organized as follows. I first describe the details of this new architecture of switching fabrics. Then a theoretical model is developed to analyze the throughput of the fabrics, which demonstrates the effectiveness of the novel architecture. Finally, we also compare it with previous parallel architectures and highlight its scalability for the next generation networks.

3.2 Proposed overflow buffer architecture in crossbar

Because our switching fabrics architecture consists of small b×2b crossbar switching elements (SEs), it is crucial to make an assumption throughout the paper that these crossbars are fixed length cell based. The cell length here is not necessarily of the same length as ATM cells (53 bytes). Other researchers have also used “packet” as term [42]. In OSI or TCP/IP model, however, “packet” always means variable length PDU (protocol data unit). In this paper we use the term “cell” as it has been done in chapter 2. From the perspective of hardware design and fair scheduler in each local SE, processing fixed length cells is much simpler and efficient than handling the variable length packets [1]. Each stage of the switching fabric can be synchronized with the same clock signal and move the cell to next stage or local outlets at the same time.

Variable length packets can be handled by segmenting them into the fix-sized cells before being transferred across the switching fabrics. At the output, the cells are reassembled into previous packets before being sent to outgoing Line Card (LC). In [42], Tzeng provides a mechanism to keep track the cells in transmission for resequencing at their destinations. So we will not cover the packet segmentation and reassemble in this dissertation.

3.2.1 Single panel multistage switching fabric

Based on research published in [40, 41], N.-F. Tzeng [42] proposed a new multistage switching fabric for scalable routers. This switching fabric, shown in Figure 3.1, is based on I-Cubeout (ICO). Each $b \times 2b$ SE has b remote outlets to connect to next stage and b local outlets to terminate the cells from the switching fabric to the destination queues. Adjacent stages are interconnected according to the indirect n-cube connecting patterns [28]. Let L be the index of line and be expressed in binary notation as follows:

$$L = 2^{n-1} l_1 + 2^{n-2} l_2 + \cdots + 2 l_{n-1} + l_n \qquad (1)$$

(Where $n = log_2 N$, N is the network size)

So the indices of the lines incident on a SE on either side differ only in l_k. Specifically, the two indices of any SE in the same stage differ by a constant; those in stage 1 differ by *N/2*, those in stage 2 differ by *N/4* and so on. A full copy of the indirect n-cube network consists of all $log_2 N$ stages. However, the ICO may contain any number of stages, but at least one full copy of stages. The stage *i* after the first full copy just repeats the (*i* mod $(log_2 N)+1$) in the same connection style as shown in Figure 3.1.

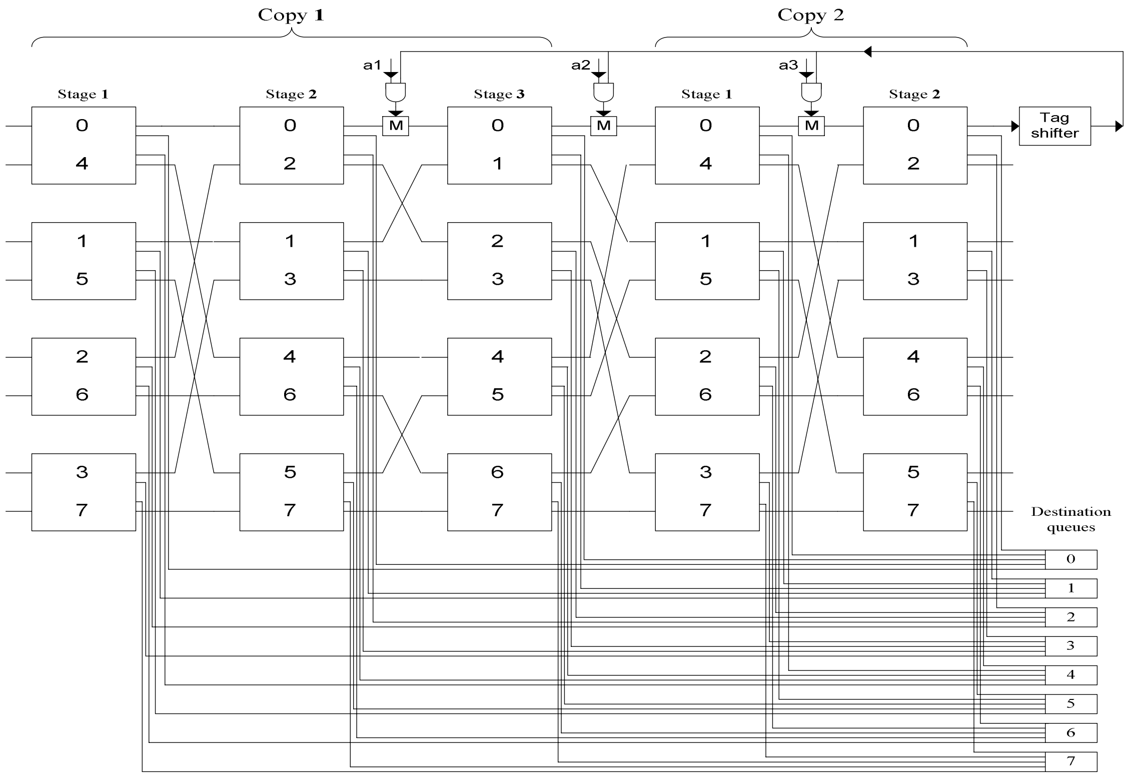

Figure 3.1: ICO_8 with recirculation.

The self-routing method of the ICO is slightly different than what is used in normal delta class network. A routing example is shown in Figure 3.2 which does not include the recirculation path and output logic for clarity. At the primary input (input of first stage), the routing tag of each cell is generated by using a bit-wise XOR of the local primary input address and its destination address. If a tag bit corresponding to stage *i* is `1`, the cell needs to take the "cross" state of SE at stage *i* of any copy. After the non-zero tag bit is corrected, it is set to `0`. If a tag bit corresponding to stage *i* is `0`, the cell just passes straight through the SE of the corresponding stage. When all the tag bits become `0`, that means the cell has reached its destined row and may take the local outlet at the SE to its destination queue, through which the output LC is connected.

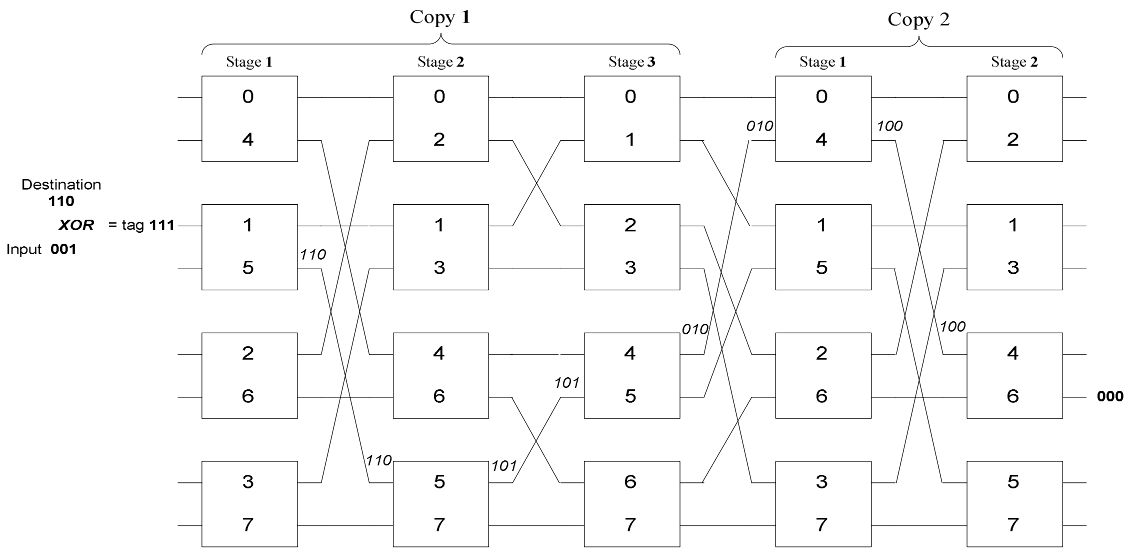

Figure 3.2: Routing of the switching fabric.

For simplicity of hardware design, the routing tag of the cell will be cyclic rotated leftward by log_2b bits after the cell advances to next stage. So, only the leftmost tag bits are examined at each SE. This will unify the design of SE without correlating it with the stage number in which the SE is located. The distance of the cell is defined as the rightmost nonzero bit position q of its tag, this means that the cell still needs to travel at least q stages before getting to its destination queue.

The local scheduler of each SE follows the shortest path algorithm. Two cells, which have different tag bit for the same SE, will conflict with each other (cross and straight through requests). The local SE scheduler should give the priority to the cell with smaller distance and deflect another one, so as to keep cells in the switching fabric as few as possible and improve the system performance. If both have identical distance, a random one is chosen for priority.

In the example shown in Figure 3.2, a cell with destination 110 comes into input 001. Then a tag 111 will be generated by the XOR operation. Next at stage 1 of copy 1, the cell cross the SE to clear the leftmost bit accordingly and cyclic shift one tag bit left. The new tag is 110. At stage 2 of copy 1, the cell is deflected and cyclic shift one tag bit left, so the corresponding bit still keeps 1. Finally, the cell arrives at destination 110 at stage 2 of copy 2 with all zero tag.

3.2.2 Recirculation connection

In order to use the switching fabric more efficiently and improve the system performance with limited hardware resources, we could reenter the cells, which failed to get to their destination queue after the primary output (the output of the last stage), into the last copy of switching fabric again by recirculation (to avoid repetitive collisions at previous copies). In [42], Tzeng proposes three approaches for choosing the reentry point shown in Figure 3.1: static connection, FA (first available point) and FO (first `1` bit in routing tag). When there is no cell arriving from the prior stage to the reentry point concurrently, the recirculated cells can be fed into the switching fabric through the multiplexers.

In [42] and the simulator used for it, the recirculation just connects back to the same physical row (physical row means the identical row in the real topology) as shown in Figure 3.3

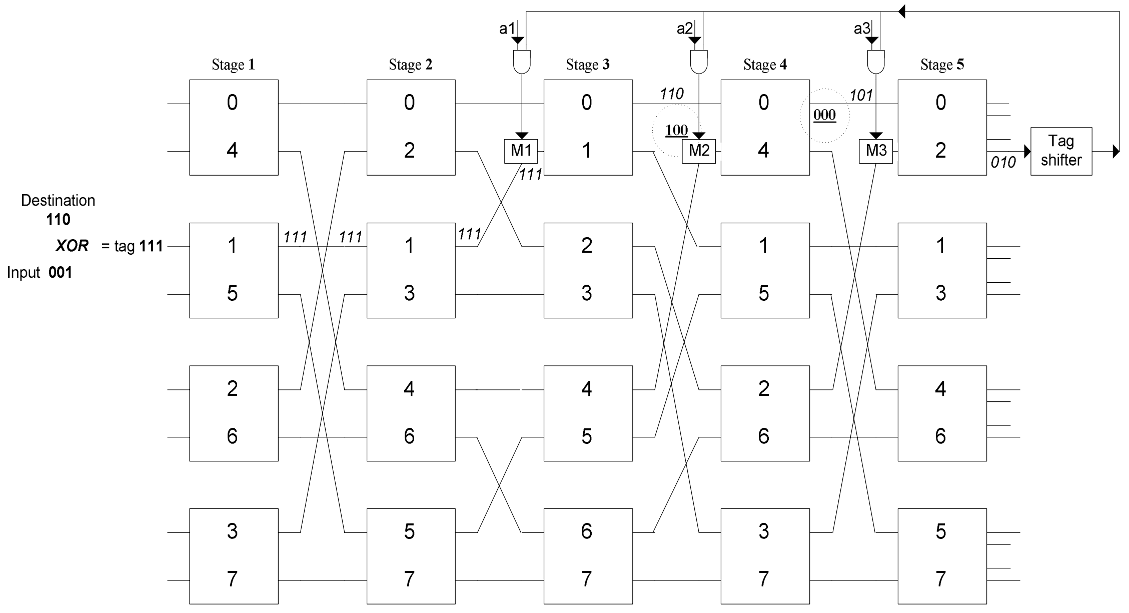

Figure 3.3: Routing failure to the same physical row.

We believe this is not correct; to show this we have a counter example in Figure 3.3. A cell with destination `110` comes into input `001`; then a routing tag `111` is generated. We assume that the cell is deflection-routed in stage 1 and 2 just as label on the path. In stage 3, the correct path is chosen, which cleared the leftmost tag bit and cyclic rotated it to rightmost position. In stage 4, the cell is deflection-routed again and chooses the correct path in stage 5. At the primary output 2, the cell will be recirculated through multiplexer M2 by FO or by FA approach (if M1 is not available). Finally, the cell is routed correctly in stage 4 and all tag bits are cleared to zero, as shown with the tags in circle. That means the cell has reached its destination queue and should be extracted from the switching fabric. But the local real address is `000` and `100`, the cell will never have a chance to reach its correct destination with a all zero tags. This routing failure is because of the recirculation to the same physical row, which changes multiple values of l_i for index in equation (1).

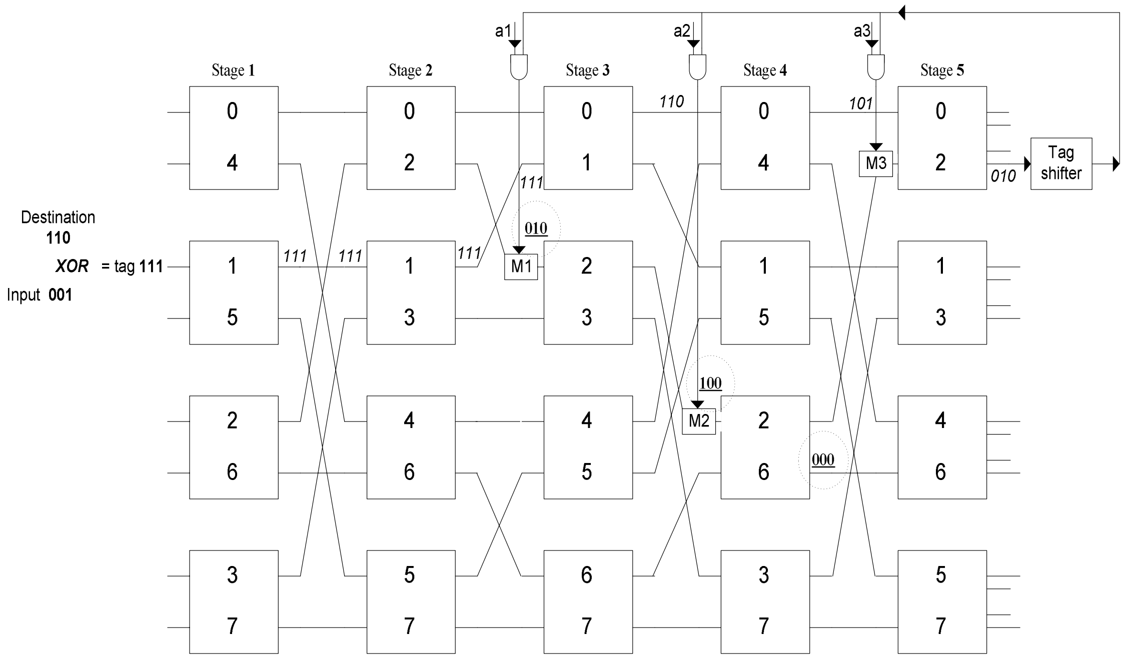

Figure 3.4: Correct recirculation to the same logic row.

So the correct recirculation should always connect to the same logic row (logical row means the same row which has identical index of lines) as shown in Figure 3.4. The cell is fed back to either M1 or M2. Finally, the cell will be terminated at the correct destination queue as shown by the tags in circle with zero tag.

3.2.3 Interleaved multistage switching fabrics

Parallel Banyan network or replicated Delta network [43] was proposed a number of years ago. But after splitting the flows at the input multiplexers, the flows will be separated independently through each panel of switching fabric. So the expensive hardware resource is sometimes not fully utilized in case of unbalanced traffic patterns. Combined with single panel of multistage switching fabric, we propose a novel architecture of interleaved multistage switching fabrics as shown in Figure 3.5.

In this architecture, multiple panels of ICO network shown in Figure 3.2 are put together by means of a multiplexer, demultiplexer and a recirculation scheme. At the N inputs, N demultiplexers distribute the input traffic into each panel synchronized with a clock. At clock cycle t, all the input cells at that time will enter the panel ((t mod Y) +1), for total Y panels from 1 to Y. At the primary outputs of each panel, we use the recirculation connections as shown in Figure 3.4 to reroute the deflection-routed cells into the switching fabric of next panel in modular. So the recirculation flows of panel i will go to panel ((i mod Y) +1). However, the recirculation entry points still follow the logic rows as shown in Figure 3.4, even though to different panels. A concentrator is located before each destination queue for terminating the cells from the local outlets of

SEs. With the speedup ξ, each concentrator could choose up to ξ cells in one clock cycle, from the active rightmost to leftmost stages independent of panels. We give higher priority to rightmost outlets in order to avoid starvation for recirculated cells in case of bursty flows.

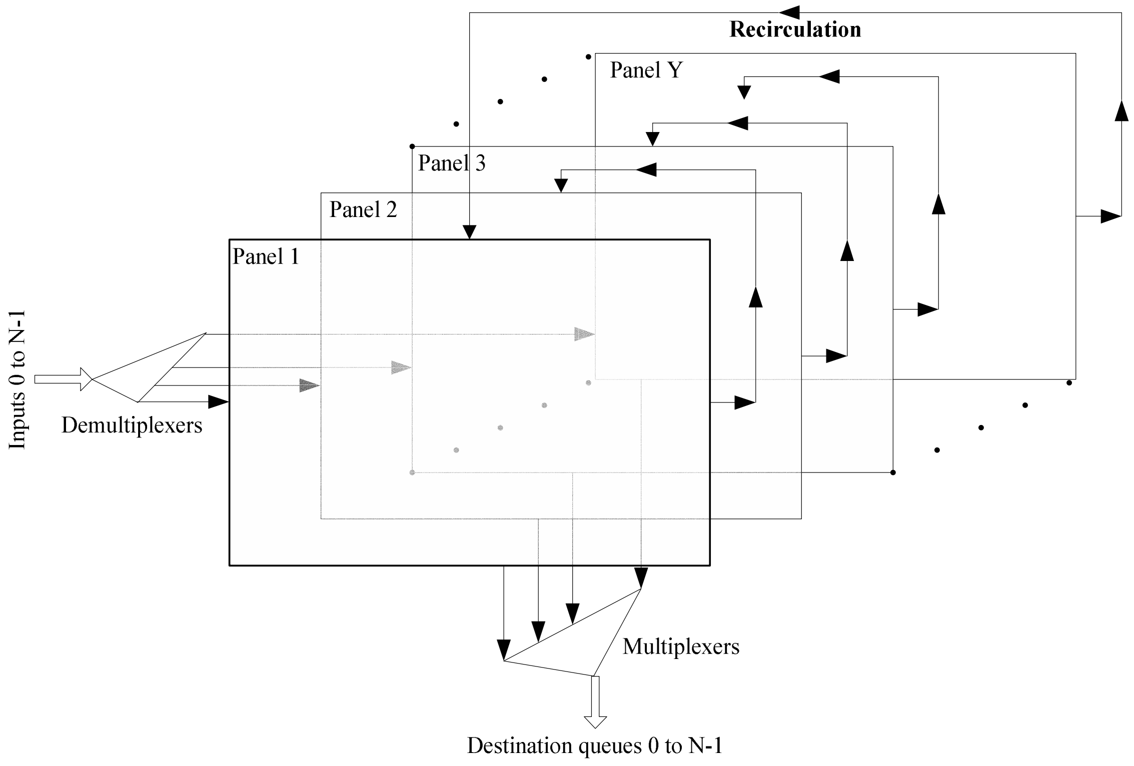

Figure 3.5: Interleaved multistage switching fabrics.

The cells take one system cycle to move from one stage to the next stage. The deflection-routed cell use one system cycle to reenter the switching fabrics through the recirculation. ICO^{FA} (first available point) and ICO^{FO} (first 1 bit in routing tag) are used in [42] to obtain better performance rather than ICO^{S} (static recirculation connection). Though ICO^{FO} is a little better than ICO^{FA} from their simulation, it needs to detect the first 1 bit from the left end in each cell's routing tag, which requires a more complicated hardware and delays the whole system. By contrast, ICO^{FA} gets the information of availability directly from prior SE's outlet latch indicator, which eases the hardware design and improves system speed. So we choose the ICO^{FA} as the recirculation approach throughout this paper.

After the multiple switching fabrics are interleaved by the recirculation, the scheme provides another opportunity to balance the cell traffic; this in turn effectively eases the hot flows after collisions. In the comparison to the single panel of switching fabric in Figure 3.1, our interleaved architecture should achieve better performance with great fault tolerance because the flows are balanced and switched in parallel. In the next section, we first analyze the new interleaved switching fabrics with a theoretical model and compare it with other traditional parallel architectures, after that we will evaluate this scheme's performance through extensive simulations.

3.3 Analytical model analysis

In last chapter 3.2, we have proposed the interleaved multistage switching fabrics. Throughput or cell drop rate (throughput = 1.0 - cell drop rate) is one of the most important parameters to evaluate a switching fabric. High performance routers require high and steady throughput under all kinds of flow conditions. Thus, an important issue is to determine the number of panels, which are enough for a real system to obtain good throughput with a reasonable hardware cost. Here, we use our analytical model under uniform traffic to address this issue; we corroborate the model's validity with simulation results.

3.3.1 Analytical model for the single panel fabric

Normally, it is very difficult to analyze the interleaved multistage switching fabrics with theoretical method if the load is non-uniform between each panel or stage. For modeling simplicity, we assume that the traffic between each panel is evenly loaded and the traffic passing from each stage to next one is uniformly distributed to each port. Moreover, we assume no buffer inside the SE between the inter-stage links. With these modeling assumptions, the complex switching system could be decomposed into each single fabric with relative independence; consequently, we just need to analyze the throughput of one of them.

We use the recursive method to get the analytical model for single panel fabric as it is used in [40, 41 and 44]. The load to the (k+1) stage is computed with the load that is not transferred to the output queues at the k stage. So if we know the random load starting at the first stage, we can compute the load to each stage through the whole fabric.

Throughout the paper, the following notations are used for the analytical model:

X : the stage number for each panel from 1 to X .

Y : the total number of panels from 1 to Y .

N : the fabric size or the input/output number from 0 to N-1.

b : the switching element size with $b \times 2b$ crossbar.

n : the stage number of one copy of the fabric， $n = \log_b N$.

P_k : the load offered to stage k, and P_{X+1} is the load for recirculation.

$q_{k,d}$:the load offered to stage k due to cells that have distance d to their destination queues.

O_k : the flow extracted from the fabric to the destination queues at stage k. Also it denotes the probability that a cell exits the fabric from stage k.

F_k : the load from the stage k to next stage. To keep the flow balance, $F_k = P_k - O_k$.

p : the load from outside the fabric, it also means the probability that a cell is generated to the input port during each cycle. So $F_0 = p$.

π: the cell drop rate of the fabric. So throughput = 1.0 - π.

With the ICO^{FA} mechanism, the cells are dropped if all the recirculation points are not available. In order to calculate the π, we need to compute the load to each stage as follows:

$$F_k = P_k - O_k \text{ and } F_0 = p. \tag{2}$$

$$P_{k+1} = F_k \text{ when } 1 \le k+1 < X - n + 1. \tag{3}$$

$P_{k+1} = F_k +$ recirculation load, when $X - n + 1 \le k + 1 \le X$ for the last copy of the fabric. To calculate P_{X-n+1} to P_X with recirculation, we use a recursive expression of the form:

$$P_{X-n+1} = F_{X-n} + (1 - F_{X-n}) \times P_{X+1}$$

$$\vdots$$

$$P_X = F_{X-1} + (1 - F_{X-1}) \times F_{X-2} \times \cdots \times F_{X-n} \times P_{X+1} \tag{4}$$

$$P_{X+1} = F_X = P_X - O_X \tag{5}$$

Thus, $\pi = P_{X+1} - (1 - F_{X-n}) \times P_{X+1} - \cdots - (1 - F_{X-1}) \times F_{X-2} \times \cdots \times F_{X-n} \times P_{X+1}$

$$= P_{X+1} \times F_{X-1} \times F_{X-2} \times \cdots \times F_{X-n} \tag{6}$$

Moreover, each P_k for $1 \le k \le X+1$ is composed of $q_{k,d}$ with:

$$P_k = \sum_{d=0}^{n-1} q_{k,d} \tag{7}$$

To evaluate (2)-(5), it is necessary to compute the O_k first. A tagged cell in stage k can exit the fabric only if its distance becomes 0 to the destination queues. Furthermore, one of the following conditions must be met:

- The tagged cell is the only one requiring a local outlet of the SE or
- More than one cell require a local outlet of the SE, but the tagged cell is chosen over the others.

Then $O_k = q_{k,0} \times [\sum_{m=0}^{b-1} \sum_{h=0}^{m} \binom{b-1}{m} \cdot \binom{m}{h} \cdot q_{k,0}^{h} \cdot (P_k - q_{k,0})^{m-h} \cdot V(h) \cdot (1 - P_k)^{b-1-m}]$ (8)

Except the tagged cell at one SE inlet, there are other $(b-1)$ SE inlets from which, h has cells that require local outlets with 0 distance, $m-h$ has cells that require remote outlets to next stage with nonzero distance, and $(b-1-m)$ have no cell for this cycle. $V(h)$ is the probability that the tagged cell is chosen in the conflict that may occur if some of the h cells require the same local outlet.

$$V(h)=\sum_{l=0}^{h}\binom{h}{l}\left(\frac{1}{b}\right)^{l}\cdot\left(1-\frac{1}{b}\right)^{h-l}\cdot\frac{1}{l+1} \tag{9}$$

To proceed the load from P_k to P_{k+1}, we need the conditional probability to compute $q_{k+1,d}$ from $q_{k,d}$ distribution.

$$q_{k+1,d}=\sum_{j=0}^{n-1}P\{q_{k+1,d}\mid q_{k,j}\}q_{k,j}\text{ , for } d=0,1,\cdots,n-1. \tag{10}$$

$P\{q_{k+1,d}\mid q_{k,j}\}$ is the conditional probability that a cell has distance *d* in stage *k*+1 after it has been switched from stage *k* where it has distance *j*. Because we use the shortest algorithm with deflection scheme in the SE's local scheduler, most of the $P\{q_{k+1,d}\mid q_{k,j}\}$ parameters are zero. Depending on the different values of *d*, three cases are distinguished as follows for (10):

1) $q_{k+1,0}=(q_{k,0}-O_k)+P\{q_{k+1,0}\mid q_{k,1}\}q_{k,1}$ (11)

2) $q_{k+1,j}=P\{q_{k+1,j}\mid q_{k,j+1}\}q_{k,j+1}$, when $0<j<n-1$. (12)

3) $$q_{k+1,n-1} = \sum_{j=0}^{n-2}[1 - P\{q_{k+1,j} \mid q_{k,j+1}\}]q_{k,j+1} \tag{13}$$

In (11), $(q_{k,0} - O_k)$ are the flows, which failed to reach their destination queues due to collisions, but they still have a zero distance to next stage with the shortest path. In (13), $q_{k+1,n-1}$ collects all the deflected flows with distance $n-1$. Because we assume that the traffic passing from each stage to next one is uniformly distributed, we could ignore the effects of $(q_{k,0} - O_k)$ to $P\{q_{k+1,j} \mid q_{k,j+1}\}$ and compute $P\{q_{k+1,j} \mid q_{k,j+1}\}$ when $0 \le j < n-1$ as:

$$P\{q_{k+1,j} \mid q_{k,j+1}\} = \sum_{m=0}^{b-1}\sum_{h=0}^{b-1-m}\binom{b-1}{m}\cdot\binom{b-1-m}{h}\cdot q_{k,j+1}^{h}\cdot(\sum_{i=1}^{j} q_{k,i})^{m}\cdot T(b-1-m-h, j+1)\cdot[1-D(m,h)] \tag{14}$$

Except the tagged cell at one SE inlet, there are $(b-1)$ other SE inlets from which, h has cells that require remote outlets with distance $j+1$, m has cells that require remote outlets with distance from 1 to j.

$T(z,i)$ is the probability that z inlets of the SE has the conditions as follows:

- Empty or
- Kept busy by cells with distance 0 or
- Kept busy by cells with distance $d > i$.

Taking into account these conditions we have that:

$$T(z,i) = \sum_{l=0}^{z}\sum_{r=0}^{z-l}\binom{z}{l}\cdot\binom{z-l}{r}\cdot q_{k,0}^{z-l-r}\cdot(\sum_{t=i+1}^{n-1} q_{k,t})^{l}\cdot(1-P_k)^{r} \tag{15}$$

$D(m,h)$ is the probability that the tagged cell is deflected, if m cells with lower distance and h cells with equal distance are switched to next stage by the SE:

$$D(m,h)=\sum_{l=0}^{m}\binom{m}{l}\cdot\left(\frac{1}{b}\right)^{l}\cdot\left(1-\frac{1}{b}\right)^{m-l}+\left(1-\frac{1}{b}\right)^{m}\cdot\sum_{l=1}^{h}\binom{h}{l}\cdot\left(\frac{1}{b}\right)^{l}\cdot\left(1-\frac{1}{b}\right)^{h-l}\cdot\frac{l}{l+1}\qquad(16)$$

From (7) to (16), we could compute the load of each stage without recirculation. For the last copy of the fabric from stage ($X-n+1$) to X , we should count in the recirculation load from the final stage $P_{X+1}=\sum_{d=0}^{n-1}q_{X+1,d}$. So, based on (4) and (11)-(13), the distributions for $0\le j\le n-1$ when $X-n+1\le k\le X$ are computed as:

$$q_{k,j}=q_{k,j}+(1-F_{k-1})\times F_{k-2}\times\cdots\times F_{X-n}\times(\sum_{i=0}^{n-1}e_{k,j,i}\cdot q_{X+1,i})\qquad(17)$$

In (17), the $q_{k,j}$ on the right hand comes from previous stage and is calculated with equations (11)-(13). $\sum_{i=0}^{n-1}e_{k,j,i}\times q_{X+1,i}$ belongs to the recirculation load. The coefficient $e_{k,j,i}$ is determined by the fabric structure N and b. For each stage k between $X-n+1\le k\le X$, the $[e_{k,j,i}]_{j\times i}$ form a 2-dimentional coefficient matrix with row j and column i , $0\le i,j\le n-1$.

As an example, when $N=256$, $b=4$ and $n=\log_4 256=4$, the four coefficient matrixes are shown below for the last 4 stages:

$$[e_{X-3,j,i}]_{j\times i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad [e_{X-2,j,i}]_{j\times i} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} \end{bmatrix},$$

$$[e_{X-1,j,i}]_{j\times i} = \begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{16} & 0 \\ 0 & 0 & 0 & \frac{1}{16} \\ \frac{3}{4} & 0 & \frac{3}{16} & \frac{3}{16} \\ 0 & 1 & \frac{3}{4} & \frac{3}{4} \end{bmatrix}, \qquad [e_{X,j,i}]_{j\times i} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & \frac{1}{64} \\ \frac{3}{4} & 0 & 0 & \frac{3}{64} \\ 0 & 1 & 0 & \frac{3}{16} \\ 0 & 0 & 1 & \frac{3}{4} \end{bmatrix}.$$

We could compute each item in matrix $[e_{k,j,i}]_{j\times i}$ from Table 3.1. Because there is no address change when cells are recirculated back to stage $X-3$, $[e_{X-3,j,i}]_{j\times i}$ is the identity matrix. Otherwise, the item $e_{k,j,i}$ depends on the recirculation point. For example, corresponding with $q_{X+1,3}$ to stage X, the third bit position A_2A_1 must be nonzero. After the cyclic rotation of the address to stage X, there are four cases as follows:

1) When $A_8A_7 = 0$, $A_6A_5 = 0$ and $A_4A_3 = 0$, the probability is $e_{X,0,3} = \dfrac{1}{64}$ for distance 0.

2) When $A_8A_7 \neq 0$, $A_6A_5 = 0$ and $A_4A_3 = 0$, the probability is $e_{X,1,3} = \dfrac{3}{64}$ for distance 1.

3) When $A_6A_5 \neq 0$ and $A_4A_3 = 0$, the probability is $e_{X,2,3} = \dfrac{3}{16}$ for distance 2.

4) When $A_4A_3 \neq 0$, the probability is $e_{X,3,3} = \frac{3}{4}$ for distance 3.

Stage \ Distance	0	1	2	3
From *X+1*	A_8A_7	A_6A_5	A_4A_3	A_2A_1
To *X-3*	A_8A_7	A_6A_5	A_4A_3	A_2A_1
To *X-2*	A_6A_5	A_4A_3	A_2A_1	A_8A_7
To *X-1*	A_4A_3	A_2A_1	A_8A_7	A_6A_5
To *X*	A_2A_1	A_8A_7	A_6A_5	A_4A_3

Table 3.1: Address cyclic rotation for recirculation with *N*=256, *b*=4.

In Table 3.1 the address cyclic rotation for recirculation is shown; in this case *N* and *b* are 256 and 4, respectively.

Based on equation (2)-(17), we could compute P_k, $q_{k,d}$ and O_k recursively until they become steady. Then with (6), the cell drop rate π (or throughput) of the single panel fabric is obtained.

3.3.2 Analytical model for the interleaved multistage switching fabrics

We have introduced an analytical model for the single panel fabric in last section and assumed that the traffic between panels is evenly loaded. Thus, each panel runs at load ($\frac{p}{Y}$), in which p is the load from outside and Y is the number of panels. The total cell drop rate (or throughput) is the sum of that from each panel. For example, when $Y = 2$ and $p = 1.0$, each panel runs at load= 0.5 and the total cell drop rate equals to 2 × (cell drop rate at load 0.5 for single panel). When $Y = 3$ and $p = 0.9$, each panel runs at load= 0.3 and the total cell drop rate equals to 3 × (cell drop rate at load 0.3 for single panel).

The benefit of the interleaved switching fabrics comes from that it avoids the high non-linear increase during load portion ($0.5 \le p \le 1$) for the single panel fabric. The interleaved architecture, which runs with load ($\frac{p}{Y} \le 0.5$) for each panel, replaces this high non-linear increase portion with linear increase of *Y* for throughput analysis. I will validate the model in next section.

3.3.3 Validation of the analytical model

The accuracy of the analytical models in chapter 3.3.1 and 3.3.2 has been assessed by comparing its results with those from simulations of the system. Figure 3.6 shows the cell drop rate under uniform traffic with parameters *N*=256, *b*=4 and *X*=4.

In the figure and below, we use the notation SX/PY where X specifies the number of stages per panel and Y the number of panels. We choose the parameters *N*=256, *b*=4 and *X*=4 because it is a full copy of a fabric which will make the results more pronounced.

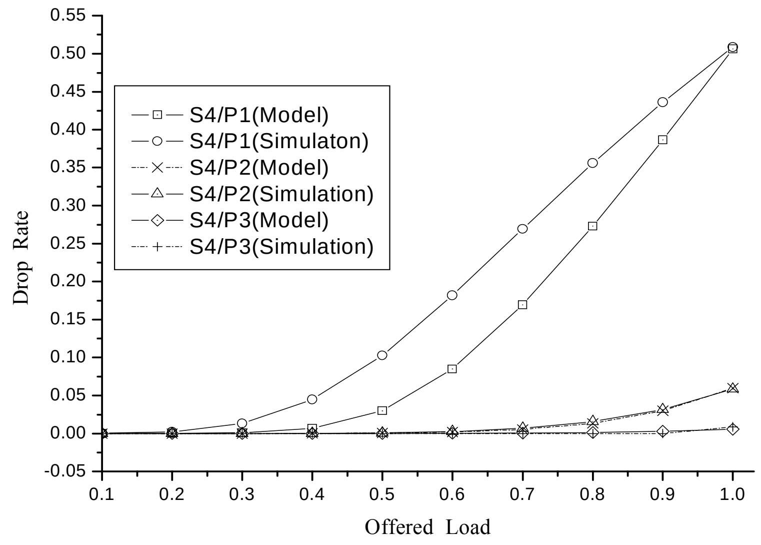

Figure 3.6: Drop rate vs offered load for analytical models and simulation with N=256, b=4 and X=4.

From Figure 3.6, there is a difference between the analytical model and simulation results for S4/P1 when load is $0.3 < p < 0.9$. As mentioned in [40, 41], the basic reason is that the traffic between the stages is unbalanced and the assumption of uniform distribution does not hold any more. When load is very low at $p < 0.3$, the traffic between stages could be still considered as uniform. When load is high enough after $p > 0.9$, all inter-stage links are saturated with traffic from previous stage or recirculation. Thus the traffic between stages could be considered as uniform again from outside view. As a result, the diagram shows satisfactory matching between models and simulation for these two load portions.

As to S4/P2 and S4/P3, the lower load to each panel, doubled or tripled recirculation path, and the interleaved connections make the traffic uniform between stages. Thus, there is a good match between the models and simulation results as shown in Figure 3.6.

To address the design issue at the beginning of Chapter 3, we need to investigate the optimal number Y of panels for good throughput over hardware cost. For Y and $Y+1$, the load to each panel is $\frac{p}{Y}$ and $\frac{p}{Y+1}$ respectively. The load difference is: $\frac{p}{Y} - \frac{p}{(Y+1)} = \frac{p}{Y(Y+1)} = \frac{p}{Y^2+Y}$, which will decrease quickly with increase of Y. That means the marginal gain of throughput will decrease accordingly with the hardware cost around $\frac{1}{Y^2}$. So $Y=2$ is the optimum number for throughput over hardware cost.

Based on the analysis above, we will choose $Y=2$ for most of the extensive simulation in the following chapters. However, $Y=3$ or more are also preferred for fault tolerant reasons as will mentioned in Chapter 5.

3.4 Comparisons with other traditional parallel architectures

A number of parallel architectures have been proposed for the high performance switching/routing systems. In this subjection we compare our interleaved switching fabrics with a few of them.

In [45], C.–T. Lea uses the multi-$\log_2$N networks in parallel to solve the internal blocking of the self-routing fabrics. However, routing algorithm of complexity O(N×$\log_2$N) is needed to dispatch each cell to a specific fabric. A much larger number of fabrics (hardware cost) are required to make the system nonblocking. These two restrictions make this architecture not scalable with N. For example, an N=256 network requires 16 fabrics for rearrangeable nonblocking. Moreover, though fault diagnosis is easy in this system, it does not provide much fault tolerance (FT) because there are no FT considerations in the routing algorithm.

In [46, 47], an innovative parallel architecture is proposed to deal with today's low speed RAM. S. Iyer and N. McKeown provide thorough analysis to provide QoS with this architecture. However, the $N \times N$ output-queued switch or CIOQ switches in the middle stage cannot support a large number of ports [46-48].

Chapter 4

Interleaved architecture performance evaluation

Although we have developed an analytical model in Chapter 3 under the assumption of uniform traffic and no buffer in the SE, we still have to use extensive simulations to evaluate the performance of our interleaved architecture with buffers in SE, under different traffic patterns and typical fault models. The typical parameters used for evaluation are the cell drop rate and mean latency which is the average time a cell takes to cross the fabrics. N.-F. Tzeng provided us with a copy of the simulator source code used in [42]; the recirculation issue shown in Figure 3.3 was corrected. The simulator was modified to fit the requirements of our architecture of interleaved switching fabrics. To have a direct comparison with results in [42], we have also chosen 256 inputs/outputs with 4×8 SEs.

4.1 Simulation model

The cell flows are fed into each panel through the 256 demultiplexers at the inputs. The input flows coming at a clock time t will go to panel ((t mod Y) +1). The offered load p

is defined as the probability that a cell is generated at each input during one cycle. In the simulations, two panels (*Y*=2) are used as mentioned in Chapter 3.3.3. If *Y*>2, the marginal gain of performance over hardware cost degrades with saturation, so we do not show the results here.

Tzeng [42] has shown that buffered fabrics will get better performance, so we also use the same model for comparison. The fabrics could run with speedup ξ. Each SE output queue (either local or remote one) is equipped with 12-cell buffers and can receive up to ξ cells during each cycle. At the outputs, each destination queue also runs with speedup ξ. Up to the capacity, ξ cells can go through the outputs to outside LC. As the simulation in [42], we have chosen $\xi= 2$.

As mentioned earlier, we choose the ICO^{FA} as the recirculation approach in our simulation. And for each result, 200,000 system clocks are simulated. This number of clock cycles is long enough to obtain steady state results.

4.2 Performance under uniform traffic pattern

Under the uniform traffic, the cells at the inputs choose each destination output with equal probability. The mean latency versus offered load under single panel *Y*=1 and interleaved double panels *Y*=2 is shown in Figure 4.1. For *Y*=1, we simulate the system with 4, 6, 8 and 12 stages. For *Y*=2, systems with 4, 6 and 8 stages per panel are simulated. As mentioned before, we use the notation SX/PY where X specifies the number of stages per panel (from 1 to X) and Y the number of panels (from 1 to Y). Thus, S8/P1 has identical number of SEs as S4/P2, just as S12/P1 with S6/P2. In terms of stages, both S4/P1 and S4/P2 (as well as S6/P1 and S6/P2, S8/P1 and S8/P2) have the same length.

From Figure 4.1, it can be observed that the interleaved fabrics significantly reduce mean latency to 4.7 cycles on Y=2 over about 16 cycles on Y=1 at load p=1.0. This in turn represents a speedup of 3.4; in this paper we use speedup as defined in Amdahl's law [49]. At load p=0.5, S4/P1 gets 4.65 cycles as the mean latency, close to that of 4.7 cycles with S4/P2 at p=1.0. So the mean latency also follows our analysis in Chapter 3.3, which avoids the high non-linear increase during load portion ($0.5 \leq p \leq 1.0$) with the interleaved architecture. Because of its parallel switching fabrics, the proposed scheme shows considerably less latency degradation with the increase of uniform traffic.

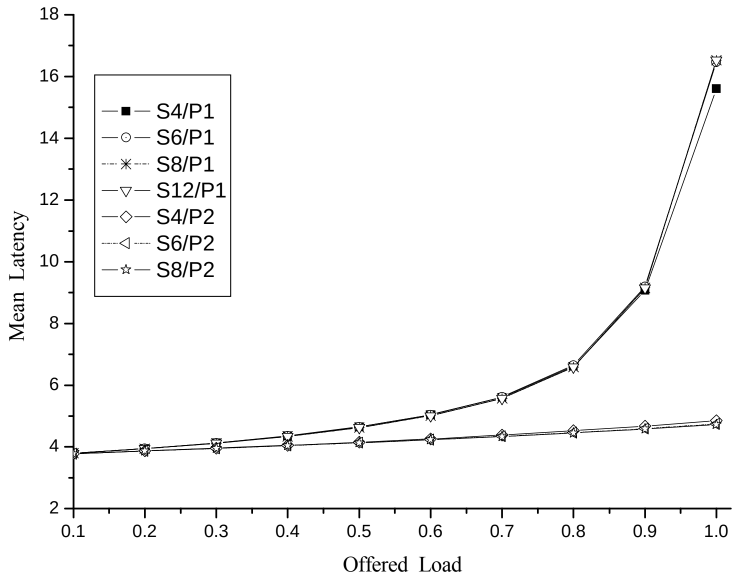

Figure 4.1: Mean latency vs offered load under uniform traffic.

Figure 4.2 shows the drop rate versus offered load. With the same length of stages, S4/P1 and S6/P1 have much larger drop rate than S4/P2 and S6/P2 respectively as predicted by the analytical models in Chapter 3; in these two comparisons the hardware is doubled. But when comparing S6/P2 and S12/P1, we find that slight increase of drop rate (0.013% of S6/P2 over 0 of S12/P1). These two configurations, S6/P2 and S12/P1, have a significant difference (about 16/4.7=3.4 times) in their mean latency; however, S6/P2 is preferred for real time connectionless applications. Even for connection-oriented applications, TCP's ARQ (Automatic Repeat Request) scheme compensates for the negligible drop rate. From another point of view, S4/P1 has 0.00581 drop rate at load p=0.5. According to the models in section 3.2, S4/P2 should have 0.00581×2=0.01162 drop rate at load p=1.0. Actually, S4/P2 has 0.01156 drop rate in simulation. The match between the values gives strong support again to our model in Chapter 3, even when we have buffers in SE.

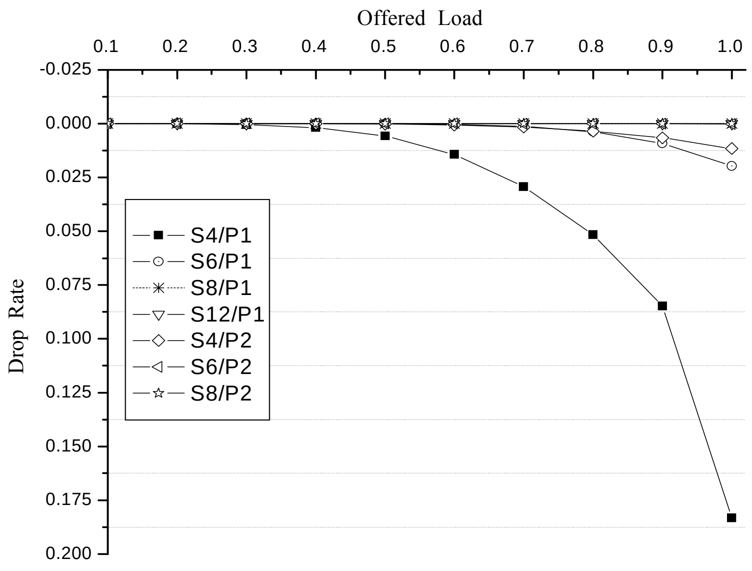

Figure 4.2: Drop rate vs offered load under uniform traffic.

The mean latency *D* shown in Figure 6 is defined as:

$$D = \frac{1}{m}\sum_{i=1}^{m} d_i \quad (18)$$

where d_i is the latency of each cell, and there is a total of m cells pass the fabrics to the outputs. So the average jitter is defined as:

$$\mathrm{J} = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(d_i - D)^2} = \sqrt{\frac{1}{m-1}[\sum_{i=1}^{m} d_i^{\,2} - mD^2]} \quad (19)$$

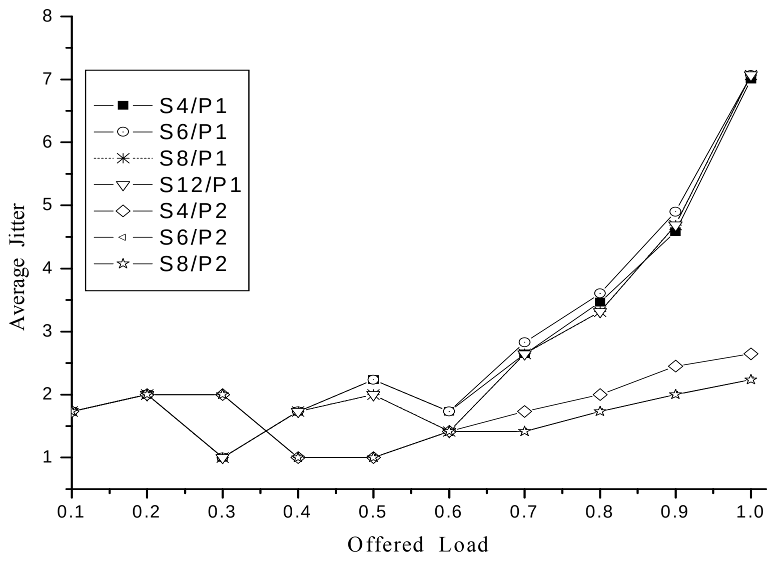

Figure 4.3: Average Jitter vs offered load under uniform traffic.

In Figure 4.3, corresponding average jitter versus offered load is shown. Average jitter at full load p=1.0 drops noticeably from more than 7 cycles of P=1 to around 2 cycles of P=2. Because the resequencing buffers at the destination queues are proportional to the average jitter of cells, the interleaved fabrics outperform the single panel fabric with fewer buffers required to assemble packets.

4.3 Performance under hot-spot traffic patterns

Traffic over the switching fabrics is usually nonuniform. There are always some hot spots on the network, such as file servers, popular web sites, and uplink to backbone network.

4.3.1 *Five hot-spots model*

Here, we first use a general five hot-spots model to measure the fabric performance under nonuniform patterns, which are shown in the following Figures. The five hot spots (around 2% of total outputs), which is chosen at output port 19, 63, 135, 182, 237, collectively receive η=12 percent hot traffic in addition to its fair share of 88 percent regular traffic left. The 88 percent regular traffic is evenly distributed over all 256 output ports. Because of the separation of the chosen output ports, the hot traffic will nearly saturate the whole fabrics.

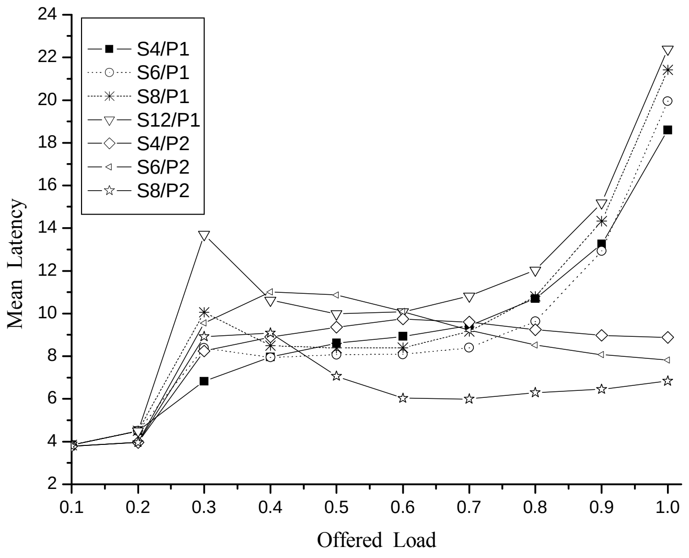

Figure 4.4: Mean latency vs offered load under 5 hot-spots traffic.

From Figure 4.4, the interleaved fabrics still exhibit great advantage with much lower mean latency against load increase (around 8 cycles of *Y*=2 over more than 18 cycles of *Y*=1 at load *p*=1.0; this means a 2.25 switching speedup). In single panel group of *Y*=1, the latency increases with more stages in each panel, because the rightmost stages have higher priority to the destination queues than the leftmost ones in Figure 5. In double panel group of *Y*=2, the results become complicated: S4/P2 has larger mean latency at p=1.0. Due to S4/P2's single copy of fabric in length, the intense congestions will deflect more cells after collision. However, deflection is expensive for the hardware resource. You need to correct the deflected tag bit back again in the following stages, which increases the overall mean latency.

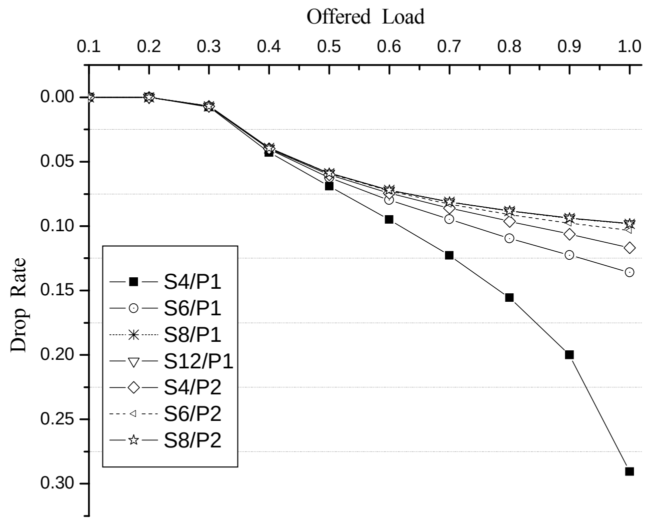

Figure 4.5: Drop rate vs offered load under 5 hot-spots traffic.

In Figure 4.5, S4/P1 still has the highest drop rate, because the recirculated cells have many collisions with the cells just from inputs. S4/P2 balances the recirculated cells between each panels and decouples them with the input flows, it decrease the drop rate from 29.1% of *Y*=1 to 11.7% at load *p*=1.0. The same phenomenon happens between S6/P1 with S6/P2. However, at *p*=0.5, S4/P1 has drop rate 0.069. 0.069×2=0.138 does not match 0.117 well of S4/P2 at *p*=1.0 because of the unbalanced hot traffic between stages, which against the assumptions in Chapter 3.

With the same hardware resource of SEs, S6/P2 just has a little higher drop rate 10.3% over 9.8% of S12/P1. Considering that S6/P2's mean latency is just 7.8 clocks opposed with S12/P1's mean latency of 22.4 clocks at *p*=1.0 (about 22.4/7.8=2.9 times), the interleaved fabrics scheme outperforms its single panel counterpart, just as it does under uniform traffic patterns. Due to ξ= 2 and minus the uniform traffic, the theoretical drop rate at full load should be: 12%(hot traffic)-2%(5hot ports/256ports)=10%. Thus, the simulation results around 10% of S12/P1 and S8/P2 match with this theoretical value.

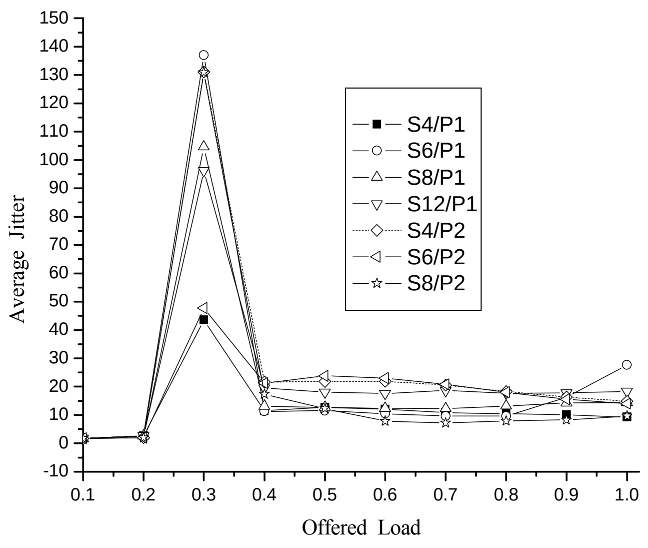

Figure 4.6: Average Jitter vs offered load under 5 hot-spots traffic.

For the average jitter, the results are shown in Figure 4.6. There is a singular point at p=0.3. From Figures 4.4 to 4.6, we could observe when the traffic is light (p<0.2), all values keep low and smooth. When traffic increase to p=0.3, the deflected hot traffic begins to saturate the fabrics. As a result, there are some glitches in Figure 4.4 and 4.6; the curves of the drop rate in Figure 4.5 bifurcate afterwards. At full load p=1.0, S6/P1 with S12/P1 still has higher average jitter than others, as same as under the uniform traffic.

4.3.2 Single hot-spot model

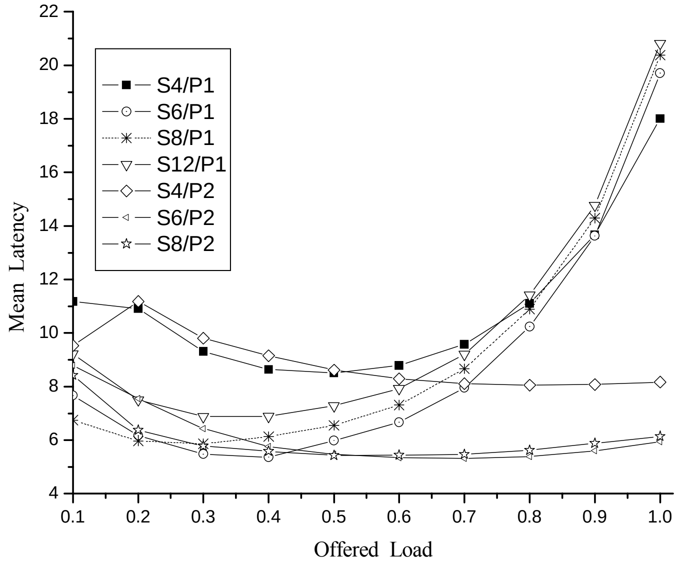

Figure 4.7: Mean Latency vs offered load under single hot-spot traffic.

In [42], Tzeng uses another extreme model with just a single hot spot to measure the fabric performance under nonuniform patterns, which should have more pronounced traffic congestions. The hot spot, which is chosen at output port 0, collectively receive η=10 percent hot traffic in addition to its fair share of 90 percent regular traffic left. The 90 percent regular traffic is evenly distributed over all 256 output ports. For comparison purpose, we also show simulations for this model in Figure 4.7 and 4.8. (The simulation for Average Jitter is similar to Figure 4.6, so we omit it here.) From Figure 4.7 and 4.8, they also show similar property of the interleaved fabrics as Figure 4.4 and 4.5, with little performance degradation against load increase.

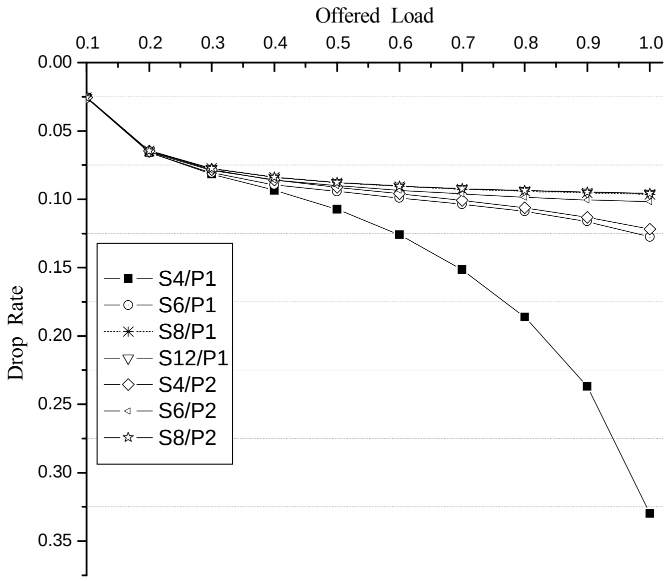

Figure 4.8: Drop rate vs offered load under single hot-spot traffic.

Chapter 5

Fault tolerance analysis

A critical design aspect of high performance routers is their reliability. Though the Internet itself is designed to tolerate failure of some router nodes, lost of core routers still results in considerable congestion to other routers with unbalanced traffic. Moreover, some subnets, which connect through the failure node, will be made unreachable.

VLSI technology moves rapidly into the nanometer range, this in turn provides faster systems and higher integration; on the other hand, the devices suffer from extreme process variation, particle-induced transient errors, and transistor wear-out. In the near future it will be unlikely to avoid having faults in VLSI systems. Considerable research has been devoted to fault tolerant switching system. In [50], the authors explore some new schemes. Though their design can tolerate a large number of defects/faults with smaller overhead than the traditional triple-modular redundancy, it still requires a considerable amount of hardware replication, which is only feasible to their 5-input CMP (chip multiprocessor) switch without much scalability.

As to our interleaved switching fabrics, its parallel architecture already has built-in redundancy that in turn provides fault tolerance. Our scheme treats a faulty element in a similar fashion as the hot congestion area, and deflects the traffic away from it. There are link and SE failures inside the switching fabrics. SE hardware is much more complex and, therefore, more prone to faults than the internal link connections. An internal link fault could also be modeled as a SE fault since a faulty link renders the following SE as a nonworkable unit. Thus in this dissertation, we use the SE fault model to evaluate its detrimental effects to system performance.

5.1 Single fault model

In the single fault model, we have a SE that is faulty which could not accept any cells. Thus, the cells that need to pass the faulty SE are deflected in prior stage. If the faulty SE is located in last copy of the fabrics, the recirculated cells need to jump this faulty point in case of the ICO^{FA} approach. However, we have observed that the stage location of the faulty SE determines the degradation to the performance other than the row location. Thus, in our simulations the faulty SE is placed at the same row but in different stages. For the results reported here we have chosen row 31 (and different stages) for symmetry purpose. Because hot–spot traffic saturates part of fabrics along its path, the effect will depend on the location of the faulty SE. Thus, we will just use uniform traffic for fault tolerant test throughout the paper.

5.1.1 Single panel and interleaved double panels

In this section, we compare how a fault impacts the performance of the single panel and interleaved double panels; both single and double panels have the same length of 6 stages.

Figure 5.1 and 5.2 show the simulation results with an increasing load in the x-axis. The fault stage locations are labeled in parenthesis.

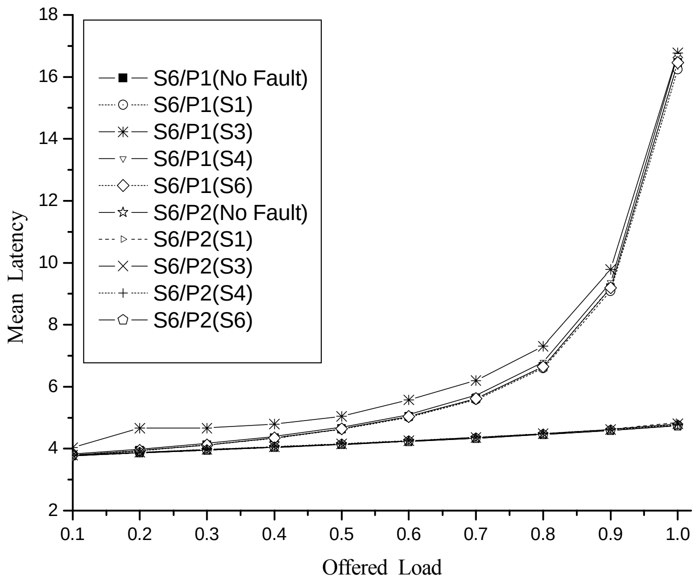

Figure 5.1: Mean latency vs offered load for single fault test (6 stages).

It is observed that the fault location determines the degradation to S6/P1 especially for drop rate. In Figure 5.2, if the faulty SE is in the first stage, the drop rate will start from 1.59%. There are total 64 SEs in each stage, 1/64= 1.56%. One faulty SE means 1.56% of the input traffic will be lost immediately without switching through the fabrics. Thus, the simulation results match the theoretical value and prove the first stage is the most critical for single panel fabric.

Another critical stage of S6/P1 is the first stage of last copy, which also merges with the first entrance point of the FA approach. If the fault happens to be this stage, both the latency and drop rate deteriorate noticeably, with a 3.23% drop rate as opposed to

1.97% for fault-free situation. Finally, the performance is insensitive to the last stage fault of S6/P1, considering most of cells have been switched to destination queues in prior stages.

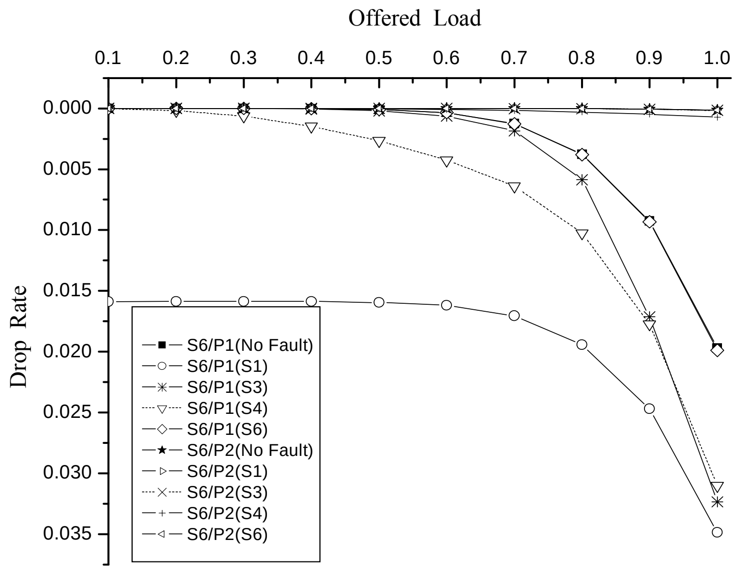

Figure 5.2: Drop rate vs offered load for single fault test (6 stages).

As expected, the faulty SE exhibits negligible impacts to performance of S6/P2 regardless of their fault locations. The interleaved fabrics in parallel not only substantially enhance the overall performance, but also tolerate the single hardware failure. Even for the fatal fault in first stage of S6/P1, S6/P2 still gives the inputs another chance to divert the flows into the fabrics.

5.1.2 Identical number of stages

In this section we compare the single and double interleaved panels when both have the same total number of stages (i.e. about the same amount of hardware resources). In this case we compare S12/P1 with S6/P2, which have the identical amount of stages. Mean latency and drop rate versus offered load are depicted in Figure 5.3 and 5.4.

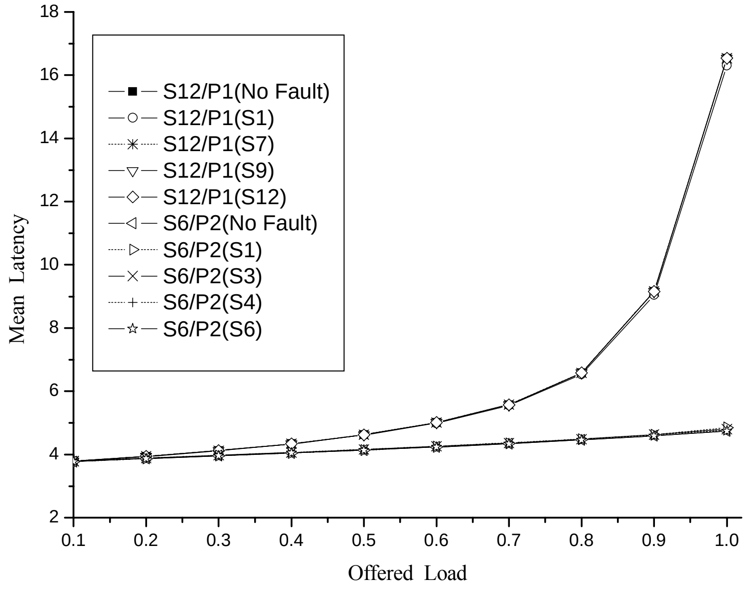

Figure 5.3: Mean latency vs offered load for single fault test (identical stages).

The simulations also indicate that S12/P1 suffers from the fatal fault of first stage just as Figure 5.2 shows. The drop rate starts at 1.59%. Thus S12/P1 will tolerate the single fault error as S6/P2 does, except for the first stage fault. However, just as Figure 4.1 and 5.3 show, the over 3 times difference in mean latency makes S6/P2 much superior over S12/P1.

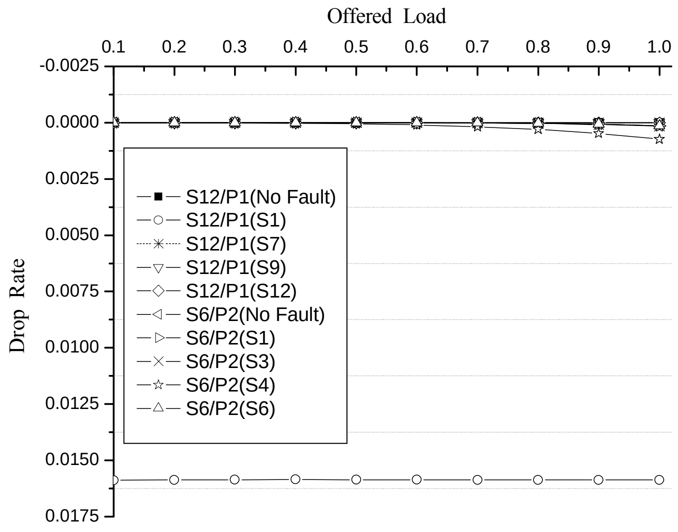

Figure 5.4: Drop rate vs offered load for single fault test (identical stages).

5.2 Multiple faults model

The multiple faults model is considerably more complicated than the single fault model, because of the abundant combinations of the number of faults and locations. However, the faults in the identical stage of different copies will generate a switching bottleneck and make the performance to deteriorate significantly, since all of them correct the same position of the tag bits. Thus, the simulations in Figures 5.5 and 5.6 depict this situation. As before, we specify the fault location in the parenthesis. For single panel, S8/P1(S4+S8) means that we choose faults at stage 4 and 8. For interleaved double panels, S6/P2[(S2+S6)/P1+S2/P2] means that we choose faults at stage 2 and 6 of panel 1 and stage 2 of panel 2. S6/P2[(S2+S6)/(P1+P2)] means that we choose faults at stage 2 and 6 for both panels, etc. Since faults have a strong tendency to happen in continuous (or nearby) areas, we assume there are continuous 4 faulty SEs in each stage and locate them at row 30,31,32,33 for symmetry purpose.

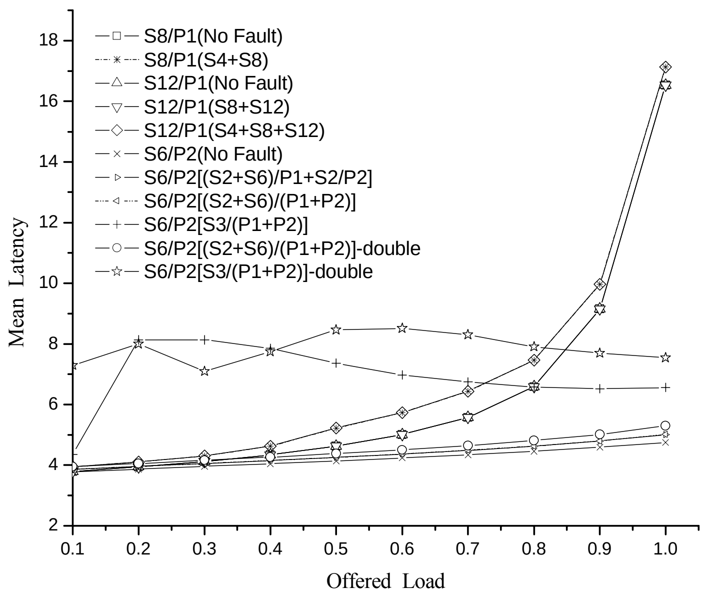

Figure 5.5: Mean latency vs offered load for multiple faults test.

As it can be observed in Figures 5.5 and 5.6, S8/P1(S4+S8) and S12/P1(S4+S8+S12) are considerably impacted by this fault model, even though S12/P1 has satisfactory performance against single fault in Figures 5.3 and 5.4. First of all, more delay latency pulls the curve above the fault-free ones. Then their drop rate jumps quickly from 9.8×10^{-8} of fault–free to 3.4% at full load p=1.0. However, the redundant stage in first copy of S12/P1(S8+S12) still provides capabilities to tolerate faults in stage 8 and 12.

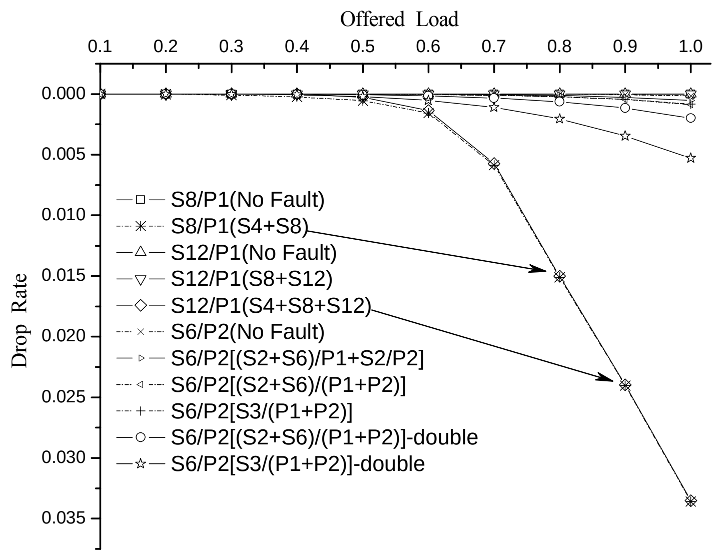

Figure 5.6: Drop rate vs offered load for multiple faults test.

For S6/P2, stages 2 and 6 of S6/P2 (total of 4 affected stages) correct the second position of the tag bits. It is reasonable that one or more redundant stages in S6/P2[(S2+S6)/P1] and S6/P2[(S2+S6)/P1+S2/P2] will compensate the faults with slight degradations in latency and drop rate. Stage 3 of S6/P2 (total of 2 affected stages) corrects the third position of the tag bits. If both of them fail, one can expect inferior performance as S8/P1(S4+S8) and S12/P1(S4+S8+S12) exhibit before. However, it is important to notice that both S6/P2[S3/(P1+P2)] and S6/P2[(S2+S6)/(P1+P2)] cause a negligible increase in drop rate, 0.086% as opposed to 0.013% of fault-free.

In order to understand why the interleaved architecture is fault tolerant, we consider first the two points where cells are dropped. One is in the SE itself when all local buffers are full. For example, if SE at row 0 stage 2 in Figure 2 fails, all flows from input 0 and 4 will merge and go through port 4 of SE at row 0 stage 1. The intense flows fill the local buffers quickly and make dropping cells unavoidable. The second point is at the FA entrance points. If all FA entrance points are not available, the recirculated cells will be dropped. Compared with the single panel architecture, S6/P2 firstly reduces the traffic to half for each panel, and then it doubles the FA recirculation points which broaden the switching path and mitigate collisions. Though correcting the deflected cells of S6/P2[S3/(P1+P2)] increases latency a little to 6.6 cycles as opposed to 4.7 cycles of fault-free as shown in Figure 5.5; this latency is still far below its counterpart of S12/P1. Moreover, with total four stages to correct the second position of tag bits, S6/P2[(S2+S6)/(P1+P2)] still achieve remarkable performance with low latency and drop rate.

Furthermore, we have even performed simulations with some extreme cases which double the faults from row 28 to 35 to a total of 8 faults in specific stages. In Figures 5.5 and 5.6, S6/P2[(S2+S6)/(P1+P2)]-double and S6/P2[S3/(P1+P2)]-double show this situation at full load *p*=1.0, with 5.3 cycles latency and 0.2% drop rate for S6/P2[(S2+S6)/(P1+P2)]-double, and 7.5 cycles latency and 0.5% drop rate for S6/P2[S3/(P1+P2)]-double. Again, the interleaved architecture shows much stronger capability of toleranting fault than its counterpart the single panel fabric.

5.3 Broken test for fabrics

In this section, we will conduct the broken test to our interleaved switching fabrics. The purpose is to test our switching system under some critical disaster conditions. We use the S6/P2 for the simulation and test the critical stage *combinations* S6/P2[(S2+S6)/(P1+P2)] and S6/P2[S3/(P1+P2)]. We will adopt *C1* to denote for S6/P2[S3/(P1+P2)] and *C2* for S6/P2[(S2+S6)/(P1+P2)] as abbreviations. We double the faulty SEs each time from 4 to 64. The faulty SEs are located symmetrically in central of each stage as before. Figures 5.7 and 5.8 depict performance deterioration under these broken tests gradually to extreme.

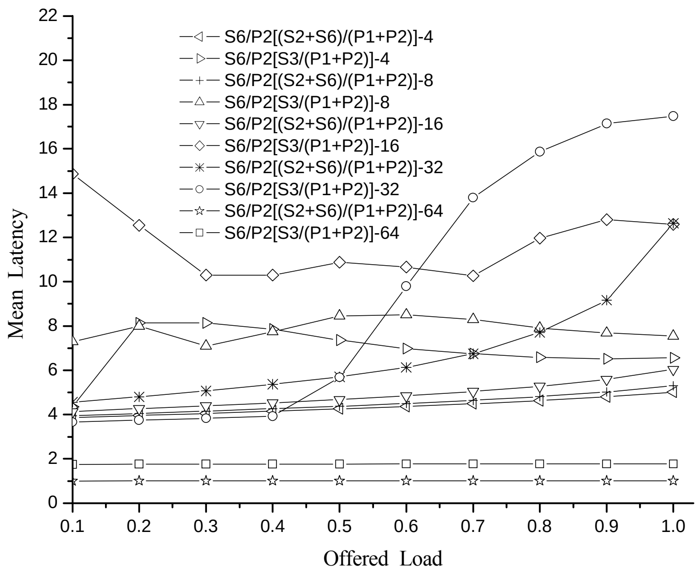

Figure 5.7: Mean latency vs offered load for broken test.

So under the worst case when 256/4=64 SEs are faulty (it means the whole stage are nonworkable), the drop rates of *C2*-64 and *C1*-64 hold 98.4% and 93.75% respectively, which match the theoretical values of (1-1/64) and (1-1/16). And the mean latency of *C2*-64 and *C1*-64 keep 1.0 and 1.77 cycles respectively, which match the theoretical values of 1.0 and [1×(1/64)+2×(1/16-1/64)]/(1/16) =1.75 cycles. When 32 SEs are faulty per stage, *C1*-32 keeps the drop rate around 46% because half of (1-1/16) flows will never have a chance to be switched on the third stage. Similar as Figures 5.5 and 5.6, *C2* still exhibit satisfactory performance even half of SEs are broken in the specific stages. For other broken test cases, our interleaved architecture shows its remarkable reliability!

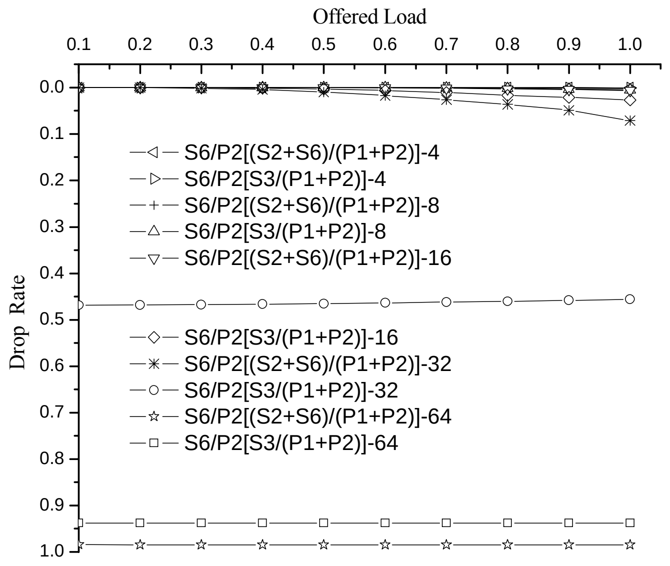

Figure 5.8: Drop rate vs offered load for broken test.

5.4 RAIF (Redundant Array of Independent Fabrics)

From the simulations and analysis above, as a good example of interleaved architecture, S6/P2 shows a better performance and much stronger capability to tolerate internal hardware failures than the single panel architecture. Specifically, each panel in S6/P2 will be a switching board which is relatively independent as mentioned in Chapter 3. Even under the worst case scenario where one panel is broken; the other panel in S6/P2 will allow the router to continue running with some performance degradation. On the other hand, in the case of S12/P1 whole system will malfunction. Moreover, inspired by RAID (Redundant Array of Independent Disks) technology [8], we could build a Redundant Array of Independent Fabrics (RAIF) by upgrading the S6/P1 with more panels running in parallel. Thus, each switching panel in RAIF works as similar as a hard disk drive in RAID system:

5.4.1 RAIF 0

Similar as RAID system level 0, the extra panels could work as RAIF 0 mode: working in parallel as S6/P2. This scheme explores all the resource available even though the marginal gain of performance decreases with more panels.

5.4.2 RAIF 1

Similar as RAID system level 1, another alternative will use additional panel working as RAIF 1 mode: stand by; this will help to lower power consumption while there is no fault and this panel will replace a malfunctioning one when fault happens. However, this scheme does not use the expensive hardware efficiently.

5.4.3 RAIF 2

Combined with RAIF 0 and 1, we could build the RAIF 2 with Y panels ($Y>2$): $Y-1$ panels working in parallel and one panel stand by for fault tolerance. In general, the

RAIF provide a flexible scalability with fault tolerance and graceful performance degradation.

5.5 Summary

In Chapter 3 to 5, we have presented a novel architecture of interleaved switching fabrics for scalable high performance routers. We also present an analytical model to assess the architecture's throughput. The benefit of the interleaved architecture comes from avoiding the high non-linear increase during load portion ($0.5 \leq p \leq 1$) for the single panel fabric.

Simulations under different traffic patterns have shown that the interleaved switching fabrics are far less sensitive on mean latency against load congestion because of its parallel switching. For the single panel fabric the mean latency deteriorates considerably as the load increases. So this property against congestion is highly preferred for backbone routers with a number of real time applications.

With the same length of stages, S4/P2 (S6/P2 and S8/P2) always outperform (in terms of latency and throughput) the single stage S4/P1 (S6/P1 and S8/P2) under different traffic conditions. The interleaved switching architecture provides speedups of 3.4 and 2.25 (or 240% and 125% improvement larger than 100%) under uniform and hot-spot traffic respectively. Although the number of SEs increases linearly, the proposed scheme shows very promising potential and scales well with present technology as compared to the exponential increase of single crossbar. We have shown that even with the same number of SEs and different organizations, S6/P2 exhibits better performance than S12/P1 (around 3 times in average latency) except for its negligibly higher drop rate. With a little increase of hardware resources, S8/P2 outperforms S12/P1 under any traffic patterns.

Moreover, our scheme treats a faulty element in a similar fashion as the hot congestion area, and deflects the traffic away from it. Extensive simulations under different faulty models have reveled that the interleaved multistage switching fabrics are highly fault tolerant against internal hardware failures that single panel fabric does not achieve. Under the worst case, the single panel fabric drops all packets when faults are located at the first stage. In general, it is possible to build a reliable, scalable high performance switching system using a Redundant Array of Independent Fabrics (RAIF) scheme in a similar fashion as RAID [8].

Chapter 6

Concluding remarks

In this chapter, the main contributions of this dissertation and potential future work are presented.

6.1 Contributions

This research encompasses a variety of architectural innovations in indirect interconnection networks for high performance routers/switches of next generation, including the following:

1. **Overflow buffer in crossbar:** Traditional crossbar design is based on symmetric buffer architecture. In this research, I have explored new asymmetric buffer architecture for the crossbar in which a new port and a local shared bus are added. Our studies indicate that we could get much improvement for the throughput and low drop rate using this new asymmetric architecture, though the latency is increased at the same time. Considering that the Internet is based on best-effort packet switching, I think that high throughout and low drop rate is much more preferred in case of the retransmission of the TCP/IP's ARQ (Automatic repeat

request) scheme. Thus we could save a lot of expensive link bandwidth and decrease the probability of congestion for the network.

2. **The RAIF scheme:** Based on the interleaved switching fabrics, I propose the novel RAIF scheme for next generation routers/switches.

   - It is scalable with hardware complexity $O(N \times \log_2 N)$ compared with $O(N^2)$ of crossbar.

   - I show how to correctly choose the recirculation points to reroute the cells back to the switching fabrics, compared with the wrong connections in former publication.

   - I present a theoretical model to assess the throughput of the interleaved switching fabrics. The benefit of the interleaved architecture comes from avoiding the high non-linear increase during load portion ($0.5 \leqslant p \leqslant 1$) for the single panel fabric. I also compare the interleaved switching fabrics with some previous parallel architectures and show effectiveness (performance and fault tolerance) of the new interleaved architecture.

   - The simulations under different traffic patterns also demonstrate its better performance of the theoretical model. The interleaved switching fabrics are far less sensitive on mean latency against load congestion because of its parallel switching. For the single panel fabric the mean latency deteriorates considerably as the load increases. So this property against congestion is highly preferred for backbone routers with a number of real time applications. With the same length of stages, S4/P2 (S6/P2 and S8/P2) always outperform (in terms of latency and throughput) the single stage S4/P1 (S6/P1 and S8/P2) under different traffic conditions. The interleaved switching architecture achieves speedups of 3.4 and 2.25 (or 240% and 125% improvement larger than 100%) under uniform and hot-spot traffic respectively at maximum load (*p*=1).

- Moreover, this new scheme treats a faulty element in a similar fashion as the hot congestion area, and deflects the traffic away from it. Extensive simulations under different faulty models have reveled that the interleaved multistage switching fabrics are highly fault tolerant against internal hardware failures that single panel fabric does not achieve. For example, the single panel fabric drops all packets when faults are located at the first stage. Even under the broken test, the interleaved architecture still exhibits satisfactory performance even half of SEs are broken in the specific stages.

- Based on the remarkable reliability and high performance achievement of the interleaved switching fabrics, I brought out the concept of Redundant Array of Independent Fabrics (RAIF). I suggest three running modes of the RAIF (RAIF 0 to RAIF 2), depending on different hardware conditions and user requirements. Hence, with this new architecture, we could build scalable, reliable, high performance routers/switches for the critical Internet infrastructure of next generation.

6.2 Future work

Figure 1.6 shows a general three-stage Clos network. When n and m equal to 2 and r is a power of 2, we could recursively replace each middle-stage switch by a three-stage network of the same structure. The resulting networks form the class of Beneš rearrangeable network as shown in Figure 6.1.

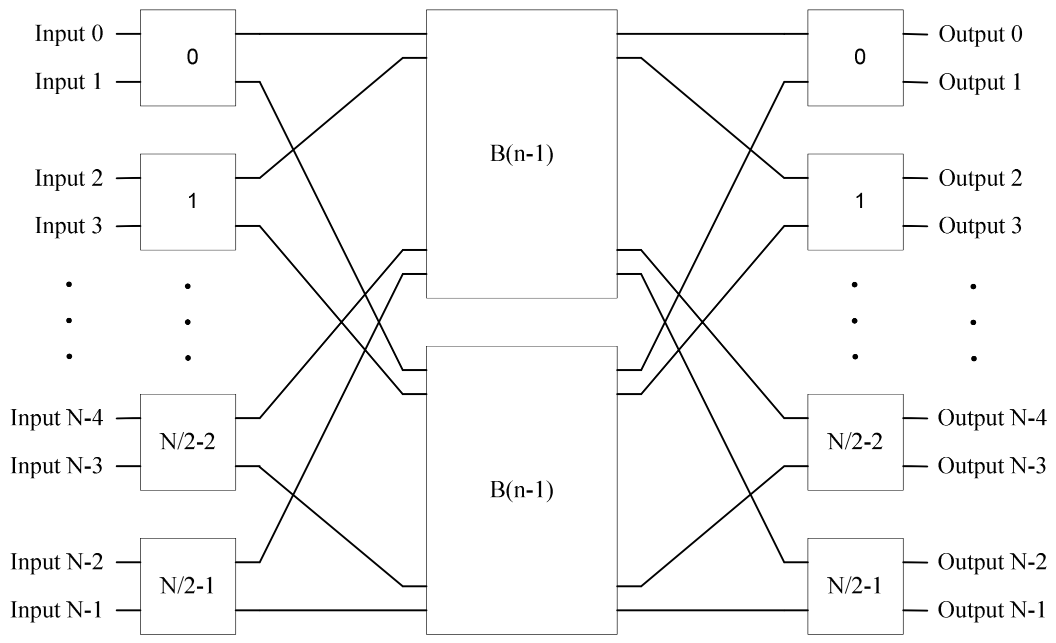

Figure 6.1: A general Beneš rearrangeable network.

6.2.1 Introduction to Beneš network

In Figure 6.1, we assume that there are N= 2^n inputs/outputs and n= $\log_2 N$. Such recursive decomposition ultimately results in a network with $2\log_2 N-1$ stages of 2×2 switches. For example, an 8x8 Beneš network is depicted in Figure 6.2.

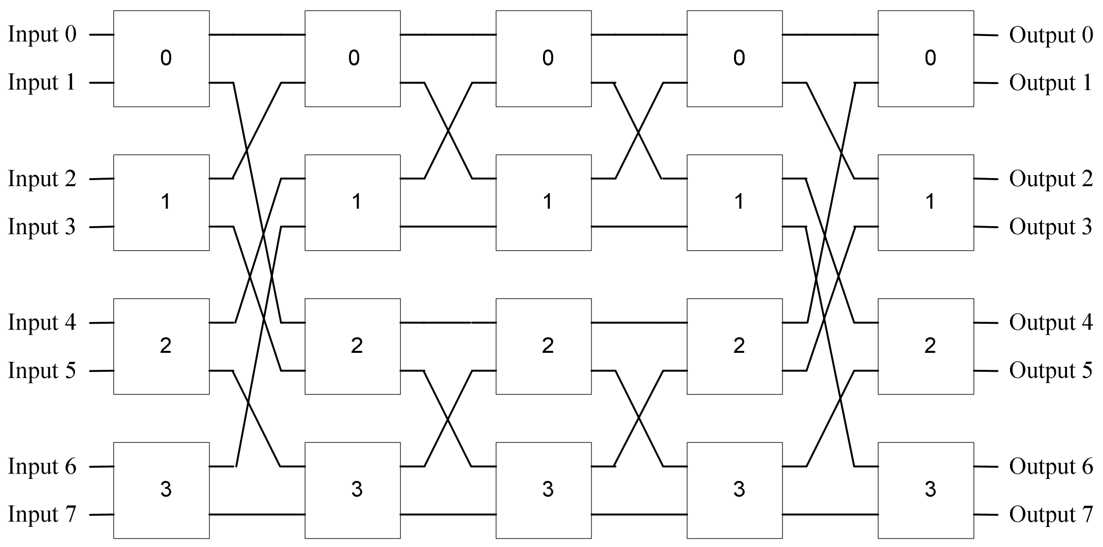

Figure 6.2: 8×8 Beneš binary network.

In [6], it has been proven that Beneš network is a permutation network with the least stages. Corresponding with recursive structure, a well-known looping algorithm [34] with complexity O(N×log$_2$N) can be used for routing in the Beneš network.

Even some parallel algorithms [35, 36] afterwards speed up the looping process with N processors. However, for these algorithms, all the 2×2 switching elements must be set up before a permutation of packets could be routed from inputs to outputs. In order to pipeline the switching stages and improve the overall throughput, a few researchers explored the self-routing approach in the Beneš network. So each stage of the switches could work independently.

In [37, 38], the author used another routing method from different point of view of Beneš network. Beside of the recursive structure, Beneš network could also be constructed with a baseline and baseline^{-1} [27] or an Omega and Omega^{-1} network [51]. The author designed a routing algorithm for the first half of the beneš network (baseline or Omega networks), so the flows could use self-routing in the second half of the beneš network (baseline^{-1} or Omega^{-1} networks). In [31], Feng proposed an inside-out routing algorithm. After configuring the two central stages, the flows will use self-routing for both sides of the baseline (Omega) and baseline^{-1} (Omega or Omega^{-1}) networks. Even though the inside-out routing algorithm is claimed to be powerful, it is found in [32] that the suggested condition of [31] for proper routing is insufficient. Other self-routing approaches in Beneš network are also proposed in [52, 53]. With the priority given to some input under collision, these approaches could implement partial of the useful permutations, not the full permutations.

6.2.2 Proposal for future research

The authors in [39] give a self-routing method with complexity $O(N\times\log_2 N)$ for full permutation. Their research is based on balanced matrix for the Omega and Omega^{-1} networks, which are equivalent to the baseline and baseline^{-1} networks. So based on this algorithm, pipelines could be implemented efficiently between each stage, which improve the overall throughput compared with [34].

However, there is still no fault tolerance consideration in this self-routing method. How to compute the tags for efficient and fault tolerant routing is still an open problem for future research.

6.2.3 Application scope

From above, I have introduced three types of indirect interconnection networks for high performance routers/switches: the crossbar, the Clos/Beneš networks and the RAIF (Redundant Array of Independent Fabrics). Because of the scalable complexity, they are feasible for different application scopes.

1. **The crossbar**: though it is internally non-blocking, the complexity increases at $O(N^2)$ in terms of crosspoint number, which become unacceptable for scalability as N becomes large. So our architecture in section 2 only adapts to the scope when **N<32**.

2. **The Clos/Beneš networks:** though they are rearrangeable networks, the routing complexity in the order $O(N \times \log_2 N)$ makes it feasible only when **N<256** with present technology. However, the fault tolerance in its self-routing method is still an open topic for future research.

3. **The RAIF scheme:** our architecture in section 3 to 5 gives a scalable solution for the scope **N>32** with simple self-routing mechanism. Though the hardware complexity is still $O(N \times \log_2 N)$, the self-routing mechanism not only avoids highly complicated algorithms, but also is feasible for simple hardware design.

Bibliography

[1] N. McKeown, "Fast Switched backplane for a Gigabit switched router," White paper, http://www.cisco.com.

[2] I. D. Scherson and A. S. Youssef, *Interconnection Networks for High-Performance Parallel Computers,* IEEE Computer Society Press, Los Alamitos, Calif, 1994.

[3] A. Varma and C. S. Raghavendra, *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice,* IEEE Computer Society Press, Los Alamitos, Calif, 1994.

[4] J. Duato, S. Yalamanchili and L. Ni, *Interconnection Networks An Engineering Approach*, IEEE Computer Society, 1997.

[5] C.Clos , "A study of Non-Blocking Switching Networks," *Bell System Technical Journal*, Vol. 32, pp. 406-424, March 1953.

[6] V. E. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.

[7] A. Kent, F. E. Froehlich, *The Froehlich/Kent Encyclopedia of Telecommunications: Volume 12 – Modernization of Telecommunications in Central and Eastern Europe to Network Management and Operations*, CRC Press, New York, 1996.

[8] D. A. Patterson, G. A. Gibson, and R. H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” *ACM SIGMOD Conf. Proc., Chicago,* pp. 109-116, June. 1988.

[9] N. Ni and L. N. Bhuyan, “Fair scheduling in Internet routers,” *IEEE Transactions on Computers*, Vol. 51, No. 6, pp. 686-701, June 2002.

[10] S. lyer, R.R. Kompella, N. McKeown, “Analysis of a Memory Architecture for fast packet buffers,” *IEEE - High Performance Switching and Routing,* Dallas, Texas, May 2001,.

[11] S. lyer, R.R. Kompella, N. McKeown, Designing packet buffers for router line cards, *Stanford University, HPNG Technical Report- TR02-HPNG-031001,* Stanford, CA, pp. 368-373, Mar, 2002.

[12] I. Keslassy, S-T.Chuang, N. McKeown, “Scaling Internet routers using optics,” *SIGCOMM’03,* August 2003.

[13] C.-S.Chang, D.-S.Lee and Y.-S .Jou, “Load balanced Birkhoff-von Neumann switches,” *Computer Comm.,* Vol 25, pp.611-622, 2002.

[14] S. lyer, R. Zhang, N. Mckeown, “Routers with a single stage of buffering,” *SIGCOMM’02*, August 2002.

[15] M.Karol, M.Hluchyj, S.Morgan, “Input versus output queuing on a space-division switch,” *IEEE Transaction on Communications,* Vol.35, pp.1347-1356, Dec 1987.

[16] Y. Tamir, G. Frazier, “High performance multiqueue buffers for VLSI communication switches,” *Proc. of 15th Annual Symp. on Computer Arch.,* June 1988.

[17] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, “High speed switch scheduling for local area networks,” *ACM Trans. Computer Systems,* Vol. 11, No. 4, pp. 319-352, Nov. 1993.

[18] N. McKeown, “The iSLIP scheduling algorithm for input-queued switches,” *IEEE/ACM Transactions on Networking*, Vol. 7, No. 2, pp. 188-200, April 1999.

[19] N. McKeown, “Scheduling algorithm for input-queued cell switches,” *Ph.D dissertation*, University of California, Berkeley, 1995.

[20] N. McKeown, A.Mekkittikul, V.Anantharam and J.Wlrand, “Achieving 100% throughput in input-queued switches,” *IEEE Transactions on Communication, Vol 47*, pp. 1260-1267, Aug, 1999.

[21] http://klamath.stanford.edu/tools/SIM/

[22] H. J. Chao, "Next generation routers," *Proceedings of the IEEE*, Vol 90, No.9, September, 2002.

[23] S. Cherry, "Noting But Net," *IEEE Spectrum,* Vol. 44, No. 1, pp. 22-26, Jan.2007.

[24] S. Jr. Ortiz, "Phone Companies Get into the TV Business," *IEEE Computer,* Vol. 39, No. 10, pp. 12-15, Oct.2006.

[25] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning processor systems," *Proc. 1st Annual Symp Computer Architecture,* pp. 21-28, Dec. 1973.

[26] D. Lawrie, "Access and alignment of data in an array processor," *IEEE Transactions on Computers*, Vol. 24, No. 12, pp. 1145-1155, Dec. 1975.

[27] C.-L. Wu and T.-Y. Feng, "On a class of multistage interconnection networks," *IEEE Transactions on Computers*, Vol. 29, No. 8, pp.694-702, Aug. 1980.

[28] M. C. Pease, "The indirect binary n-Cube microprocessor array," *IEEE Transactions on Computers*, Vol. 26, No. 5, pp. 458-473, May 1977.

[29] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Transactions on Computers*, Vol. 30, No. 10, pp. 771-780, Oct. 1981.

[30] C.-L. Wu and T.-Y. Feng, "The universality of the shuffle-exchange network," *IEEE Transactions on Computers*, Vol. 30, No. 5, pp. 324-332, May 1981.

[31] T.-Y. Feng and S.-W. Seo, “A new routing algorithm for a class of rearrangeable networks,” *IEEE Transactions on Computers*, Vol. 43, No. 11, pp.1,270-1,280, Nov. 1994.

[32] M. K. Kim, H. Yoon, and S. R. Maeng, “On the correctness of inside-out routing algorithm,” *IEEE Transactions on Computers*, Vol. 46, No. 7, pp. 820-823, July. 1997.

[33] H. Çam, “Rearrangeability of (2n-1)-stage shuffle-exchange networks,” *SIAM J. COMPUT*, Vol. 32, No. 3, pp. 557-585, 2003.

[34] D. C. Opferman and N. T. Tsao-Wu, “On a Class of Rearrangeble Switching Networks, Part I: Control Algorithm,” *Bell System Technical Journal*, Vol. 50, No. 5, pp. 1579-1600, May-June 1971.

[35] G. Lev, N. Pippenger, and L. G. Valiant, “A fast parallel algorithm for routing in permutation networks,” *IEEE Transactions on Computers*, Vol C-30, No. 2, pp. 93-100, Feb. 1981.

[36] D. Nassimi, and S. Sahni, “Parallel algorithm to set up the Beneš permutation network,” *IEEE Transactions on Computers*, Vol C-31, No. 2, pp. 148-154, Feb. 1982.

[37] K. Y. Lee, “On the rearrangeability of 2($\log_2$N)-1 stage permutation networks,” *IEEE Transactions on Computers*, Vol C-34, No. 5, pp. 412-425, May. 1985.

[38] K. Y. Lee, “A new Beneš network control algorithm,” *IEEE Transactions on Computers*, Vol C-36, No. 6, pp. 768-772, June. 1987.

[39] H. Çam and J. A.B. Fortes, “Work-Efficient Routing Algorithms for Rearrangeable Symmetrical Networks,” *IEEE Transactions on Parallel and Distributed Systems,* Vol. 10, No. 7, pp. 733-741, July 1999.

[40] S. Bassi, M. Decina, P. Giacomazzi, and A. Pattavina, “Multistage shuffle networks with shortest path and deflection routing for high performance ATM switching: The open-loop Shuffleout,” *IEEE Transactions on Communication,* Vol. 42, No. 10, pp. 2881-2889, Oct.1994.

[41] M. Decina, P. Giacomazzi, and A. Pattavina,, “Multistage shuffle networks with shortest path and deflection routing for high performance ATM switching: The closed-loop Shuffleout,” *IEEE Transactions on Communication,* Vol. 42, No. 11, pp. 3034-3044, Nov. 1994.

[42] N.-F. Tzeng, “Multistage-based switching fabrics for scalable routers,” *IEEE Transactions on Parallel and Distributed Systems,* Vol. 15, No. 4, pp. 304-318, April 2004.

[43] C. P. Kruskal and M. Snir, “The performance of multistage interconnection networks for multiprocessors,” *IEEE Transactions on Computers*, Vol 32, No. 12, pp. 1091-1098, Dec. 1983.

[44] A. Pattavina, *Switching Theory: Architecture and Performance in Broadband ATM Networks,* Wiley, 1998.

[45] C.–T. Lea, “Multi-log$_2$N networks and their applications in high-speed electronic an photonic switching systems,” *IEEE Transactions on Communication,* Vol. 38, No. 10, pp. 1740-1749, Oct. 1990.

[46] S. Iyer and N. McKeown, “Analysis of the parallel packet switch architecture,” *IEEE/ACM Transactions on Networking*, Vol. 11, No. 2, pp. 314-324, April 2003.

[47] S. Iyer and N. McKeown, “Making parallel packet switches practical,” *IEEE INFOCOM,* Alaska, USA, 2001, Vol. 3, pp. 1680-1687.

[48] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar “Matching output queueing with a combined input/output-queued switch,” *IEEE J. Select. Areas Commun,* Vol. 17, No. 6, pp. 1030-1039, June 1999.

[49] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach,* 3rd Edition. Morgan Kaufmann, 2003.

[50] K. Constantinides S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, “BulletProof: A Defect Tolerant CMP Switch Architecture,” *International Symposium on High-Performance Computer Architecture (HPCA),* pp. 5-16, Feb. 2006.

[51] Y.-M. Yeh and T.-Y. Feng, "On a class of Rearrangeable Networks," *IEEE Transactions on Computers*, Vol. 41, No. 11, pp.1361-1379, Nov. 1992.

[52] D. Nassimi, and S. Sahni, "A self-Routing Beneš Network and Parallel Permutation Algorithms," *IEEE Transactions on Computers*, Vol C-30, No. 5, pp. 332-340, May. 1981.

[53] C. S. Raghavendra, and R. V. Boppana, "On self-Routing in Beneš and Shuffle-Exchange Networks," *IEEE Transactions on Computers*, Vol 40, No. 9, pp. 1057-1064, Sep. 1991.

Appendix

Publications

Portions of this dissertation appear in the following publications:

[1] R. He and J. Delgado-Frias, “Overflow buffer in crossbar- A new architecture for high performance routers,” *The 2nd IASTED International Conference on Communication and Computer Networks*, 2004.

[2] R. He and J. Delgado-Frias, “Interleaved multistage switching fabrics for scalable high performance routers,” *Proceedings of* ***49th IEEE GLOBECOM*** *conference*, San Francisco, CA, Nov, 2006.

[3] R. He and J. Delgado-Frias, “Fault tolerant interleaved switching fabrics for scalable high performance routers,” **accepted** by ***IEEE Transactions on Parallel and Distributed Systems***.

[4] R. He and J. Delgado-Frias, “Redundant Array of Independent Fabrics – An Architecture for Next Generation Network,” **accepted** by ***50th IEEE GLOBECOM*** *conference,* Washington, D.C, Nov, 2007.