A Practical Scheduler For High-Speed Packet Switches and Internet Routers

Lotfi Mhamdi   Stamatis Vassiliadis
Computer Engineering Lab, Delft University of Technology,
P.O. Box 5031, 2600 GA Delft,The Netherlands
E-mail: lotfi@ce.et.tudelft.nl

Abstract—**The input queued (IQ) crossbar based switching, employing virtual output queueing (VOQ), is the dominant architecture for high-performance packet switches. The performance of a VOQ switch depends solely on the scheduling algorithm used. Maximum Weight Matching (MWM) algorithms have optimal performance however they are not practical due to their hardware complexity. Round Robin (RR) based algorithms, on the other hand, are practical but lack performance. The goal of this paper is to propose a scheduling algorithm that has the performance of the MWM based algorithms and the cost and reduced complexity of the RR based algorithms. We show that, by combining the Static Round Robin (SRR) technique with a *pre-Matching* (pM) phase, we increase the performance. In order to mimic the performance of the MWM algorithms, we give priority to newly arriving traffic. In particular, we propose a new SRR-pM based scheduling algorithm named SMART (*Srr-pM current ARrival firsT*). Our proposed algorithm achieves high-performance under a wide range of realistic traffic patterns. Furthermore, SMART requires no extra hardware complexity compared to state-of-the art RR algorithms while achieving higher performance.**

Keywords— **Switching, Scheduling, VOQ, Static Round-Robin, Performance, Hardware requirement.**

I. Introduction

The crossbar-based fabric switching is the dominant architecture in today's high-performance packet switches (IP routers, ATM switches, Ethernet switches). In simplest terms, a packet switch is a store (queuing) and forward (switching) network device. Depending on when and where the queuing and switching is performed, the architecture of a packet switch can be: a) Output Queued (OQ), where the queuing takes place after the switching or at the output side; b) Input Queued (IQ), where the queuing function is performed before the switching or at the input side; c) Combined-Input-Output-Queued (CIOQ), where the queuing takes place in the input as well as the output side with the switching function is in the centre.

OQ switching is widely considered as the ideal packet switch architecture due to its optimal performance and QoS guarantees [1][2]. For an $N \times N$ OQ switch, the memory buffers need to run $N+1$ faster than the line rate. Unfortunately the limitation in memory access time coupled with the high internal speedup prohibits the OQ architecture to be practical for even a medium sized switch. Therefore, this architecture has been used only as a reference model to assess the performance of other architectures. IQ switches, on the other hand, have been considered as the lowest cost solution with highest scalability. The IQ switch has a low internal speed up because the crossbar fabric has the same speed as that of the external line. Unfortunately, when the first-in-first-out (FIFO) queueing discipline is used at the input queues, the IQ cannot achieve more than 58.6% throughput due to the Head-of-Line (HoL) blocking problem [3].

The HoL blocking can be completely eradicated by using virtual output queuing (VOQ) architecture at the input side [4][5]. Rather than maintaining a single FIFO queue for all cells, each input maintains a separate queue for each output as shown in Figure 1. Thus, there are a total of N^2 input queues. Each separate queue is called a VOQ and operates according to the FIFO discipline. The VOQ overcomes the HoL problem and scales up the achievable throughput of a switch to 100%, making the IQ switch even more attractive. Adopting a VOQ switch, a centralized scheduler is required to avoid input and output contention and configures the switch accordingly. At each time slot, the scheduler has to find a conflict free match between the inputs and the outputs. As a result, the switching performance essentially depends on the scheduling algorithm [6].

A wide range of scheduling algorithms have been proposed in the literature. These algorithms can be broadly classified into two main categories. Maximum Weight Matching (MWM) and iterative algorithms that approximate the MWM algorithms. Despite the optimal performance that MWM algorithms exhibit [7] [4], they are considered unattractive due to their hardware and time complexity. As a result, many practical iterative algorithms that approximate the MWM algorithm have been proposed, such as PIM [8], iSlip [9], DRRM [10], FIRM [11] and SRR [12]. These algorithms are based on a simple three phases hand-shaking scheme known as *Request-*

Grant-Accept (RGA). They are simple in hardware and can achieve fairness due to their round robin nature. Unfortunately, none of these algorithms reach the performance of an OQ switch or the MWM algorithm.

In this paper, we propose a new algorithm designed to overcome the weakness of the round robin based algorithms while kept readily implementable in hardware. We first study the properties of the Static Round-Robin (SRR) technique [12] and apply it to a wide range of scheduling algorithms. We show that, by introducing a simple *"pre-Match"* (pM) step, we can dramatically improve the performance of the scheduling algorithms at no extra hardware cost. In particular, we propose a new SRR-pM based scheduling algorithm called SMART (*Srr-Mf current ARrival firsT*). SMART gives priority to the most recently arriving packets to the switch. This implies that the input ports with heavier traffic gain a higher chance to be served first, whereas when the traffic is uniform all the ports have the same scheduling priority. In this way, SMART overcomes the performance limitations of previous RR based algorithms by effectively handling both uniform and non-uniform traffic flows. This makes SMART an algorithm with the performance of MWM algorithms at the cost of RR based algorithms. The simulation results showed that our proposed algorithm achieves high-performance under a wide range of realistic traffic patterns. Furthermore, SMART requires no extra hardware complexity compared to state-of-the-art algorithms while running at very high speed.

The remainder of the paper is organized as follows: In Section II, we analyze the performance of existing round-robin algorithms. Section III details our proposed algorithm along with its properties. Section IV outlines the performance evaluation of our algorithm with a comparison to state-of-the-art algorithms. Finally, Section V concludes the paper.

II. Existing Scheduling Algorithms

Most high-performance switches and Internet routers are built based on a crossbar switch with a centralized scheduler as depicted in Figure 1. Variable length packets are segmented into fixed-size cells upon their arrival to the switch and reassembled before they leave the switch. The centralized scheduler considers the current occupancy of all the VOQs and finds a configuration of the crossbar fabric matrix such that, each time slot, every input can send at most one cell and each output can receive at most one cell. Cells arriving at input i and destined for output j are queued in $VOQ_{i,j}$. In order to approximate the MWM schemes, maximal size matching schemes (MSM) were proposed. This class of scalable schedulers is based on the RGA handshaking scheme. The PIM (Parallel Iterative Matching) was the first proposed algorithm using this scheme [8]. It uses randomness to avoid starvation and reduce the number of iterations needed to converge to a maximal-sized match. However, this scheme proved unattractive because the random choices are costly and slow in terms of performance.

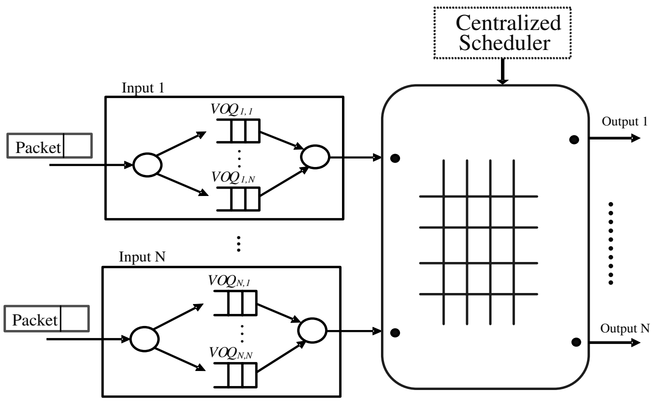

Fig. 1. The VOQ Architecture

A. Round-Robin Scheduling

The RRM (Basic Round-Robin Matching) overcomes the complexity and unfairness of PIM by using simple round-robin arbitration. It is the first proposed algorithm that is based on the round robin discipline. The round-robin arbiter can be readily implemented in hardware by priority encoders and can operate at very high speed. However, the throughput of RRM under uniform traffic is limited to 63% due to the pointers synchronization effect [4]. The specification of RRM is as follows:

- `Step 1:` *Request.* Each input sends a request to every output for which it has a queued cell.
- `Step 2:` *Grant.* If an output receives any requests, it chooses the one that appears next in a fixed round robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer g_i to the highest priority element of the round-robin schedule is incremented $(modulo\ N)$ to one location beyond the granted input.
- `Step 3:` *Accept.* If an input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The pointer a_i to the highest priority element of the round-robin schedule is incremented $(modulo\ N)$ to one location beyond the accepted output.

B. The iSlip Algorithm

The iSlip scheduling algorithm is, probably, the most known VOQ scheduling scheme nowadays. It is similar to the RRM algorithm and was first proposed in [9]. The iSlip overcomes the desynchronization problem by making a slight, yet very significant, change into the pointer updating scheme. In step 2, the grant pointer gets updated *if and only if* the grant is accepted by the input. It has the following specification:

- `Step 1`: *Request.* Same as RRM.
- `Step 2`: *Grant.* If an output receives any requests, it chooses the one that appears next in a fixed round robin schedule, starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented $(modulo\ N)$ to one location beyond the granted input *if and only if the grant is accepted in Step 3.*
- `Step 3`: *Accept.* Same as RRM.

The FIRM (FCFS In Round-Robin Matching) [11] goes one step further taking care of its pointer updating scheme. With FIRM, the grant pointer is set to the granted position even if the grant is not accepted in Step 3. The scheduling process of iSlip is depicted in Figure 2.

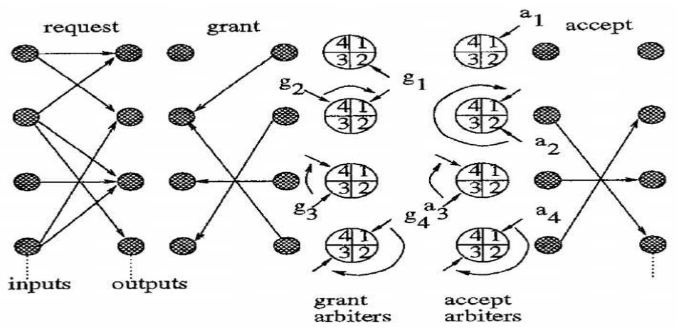

Fig. 2. The iSlip Scheduling Algorithm

C. The Dual Round-Robin Matching

The DRRM (Dual Round-Robin Matching) algorithm simplifies the iSlip algorithm into 2 steps [10]. The secret of removing one step is that each input makes only one grant to the outputs instead of sending out all the requests. Likewise, each output accepts only one request to be granted. The specification of DRRM is the following:

- `Step 1`: *Request.* Each unmatched input, i, selects a non-empty $VOQ_{i,j}$ in a round-robin fashion starting from the pointer a_i and sends the request to the output j.
- `Step 2`: *Grant.* If an output, j, receives any request, it grants the input in a round-robin fashion starting from the pointer g_j.

Because the pointer updating mechanism is the key difference among these algorithms, we plot the pointer's updating scheme for each of iSlip and FIRM, or their combination in table I. Similarly, FIRM can be applied to DRRM

TABLE I
POINTER UPDATE SCHEME FOR ISLIP AND FRIM

		iSlip	iSlip-FIRM
g_j	No Request	unchanged	
	Grant Accepted	one location after the granted one	
	Grant Unaccepted	unchanged	granted one
a_i	No Grant	unchanged	
	Grant Accepted	one location after the accepted one	

TABLE II
POINTER UPDATE SCHEME FOR DRRM AND FRIM

		DRRM	DRRM-FIRM
r_i	No Request	unchanged	
	Request Granted	one location after the requested one	
	Request Un-granted	unchanged	requested one
g_j	No Request	unchanged	
	Granted	one location after the granted one	

too. The pointer update schemes for DRRM and FIRM are summarized in Table II. While each of these algorithms uses a different pointer updating scheme, they all share the objective to completely avoiding the pointers synchronization problem. In the following section we will present the Static Round Robin (SRR) scheme that, unlike all others, succeeded as a fully desynchronized round robin scheme.

D. The Static Round Robin Matching

The Static Round-Robin (SRR) scheme was firstly introduced by [12]. The key idea of SRR is to artificially desynchronize the arbitration pointers at the beginning and, as time goes on, they get updated in a static way to keep them continuously desynchronized. SRR uses exactly the same handshaking scheme as in the in iSlip or DRRM. The only difference with iSlip is in the way of initializing and updating the pointers as summarized in Table III.

TABLE III
UPDATE SCHEME FOR SRR IN ISLIP

a_i		g_j	
Initialization	Update	Initialization	Update
$a_i = i$	increment by one	$g_j = j$	decrement by one

The round-robin searching direction is defined with respect to the pointer update sequence. A Variant called *Clockwise* direction was introduced in SRR, meaning move the pointer in the same direction as the pointer update sequence. For example, if the pointer is incremented after a match, the Clockwise direction will search incrementally starting from the pointer's location. The reason for this is, unlike iSlip where the pointers are updated only if there is a match and therefore it is better if we always search in the Clockwise direction, SRR always update the pointer even though there may be no match. This may cause unfairness when the traffic is unbalanced. As a result, SRR has been designed to search in a Clockwise and *Anti-clockwise* direction alternatively in each time slot.

III. The SMART Scheduling Algorithm

In this section, we introduce our proposed scheduling algorithm. SMART is designed as a round-robin based scheduler and expected to compete with the performance of the MWM algorithms. Based on the analysis of the previous round-robin based schedulers above, we learned that the pointer updating mechanism is of key importance. We chose the SRR updating scheme due to its optimal pointers desynchronization. Further more, and in order to further improve the performance, we introduce a *pre-Mtach* step in SMART. The reason for this step is to optimize the communication between the grant and accept arbiters and make it more effective. For example if input 1 has a queued cell, c, in $VOQ_{1,3}$, and its accept pointer, a_i, is pointing to output 3, we know that cell c is guaranteed to be selected by the scheduler due to the desynchronizing effect of SRR. Therefore, we set input 1 to be matched to output 3 during the *pre-Match* step. In other words, if an input accept pointer is pointing to a non empty $VOQ_{i,j}$, the *pre-Mtach* step will set input i and output j to be matched affording more chance to other requests to be granted and other grants to be accepted.

In order to mimic the performance of the MWM algorithms, we added extra intelligence in the scheduling process. We give priority to newly arriving cells to the switch. The intuition behind this is to overcome the lack of performance under the non-uniform traffic without using any weight functions or state information. The idea of serving the newly arriving cells favors the input that has a greater frequency of arriving cells without punishing the uniformly arriving cells, hence tackling the non-uniform traffic while being stateless. The specification of SMART is as follows:

- `Step 0:` *Match.* For each input i, if $VOQ(i, a_i)$ is non-empty, set input i and output a_i to be matched.
- `Step 1:` *Request 1.* For each unmatched input i, if it has a newly arrived cell destined to output j in current time slot, send the request to output j.
- `Step 2a:` *Grant 1.* If an unmatched output j receives one or more request, it selects the one that appears next in a fixed round-robin fashion starting from pointer g_j. The output notifies the granted input.
- `Step 2b:` *Request 2.* Simultaneously, each unmatched input i selects a $VOQ_{i,j\prime}$ from all non-empty VOQs, except one with the newly arrival cell, in a fixed round-robin fashion starting from pointer a_i. If the input receives the grant from output j in step $2a$:
  - Set input i and output j matched.
  - Else, send a request to output $j\prime$.
- `Step 3:` *Grant 2.* If an unmatched output $j\prime$ receives one or more request, it selects the one that appears next in a fixed round-robin fashion starting from pointer $g_{j\prime}$. The output notifies the granted input, and sets both to be matched.

IV. Performance Study

This section presents a delay performance study using a 32×32 IQ crossbar switch with VOQ structure. The delay is measured as the period of time a cell is kept waiting in an input buffer before being scheduled. Each point in the resulting figures is obtained for 500,000 time slots (cell time), and the statistics are gathered from the $(50,000)^{th}$ time slot. Average cell delay is calculated from all the cells output during this period of time. Normalized load means the percentage of time slots which have cells coming in, averaged over all inputs. The SMART algorithm is compared to the Islip, SRR and to the optimal OQ switch. The traffic patterns used are Bernoulli I.I.D. uniform, bursty uniform traffic and unbalanced traffic.

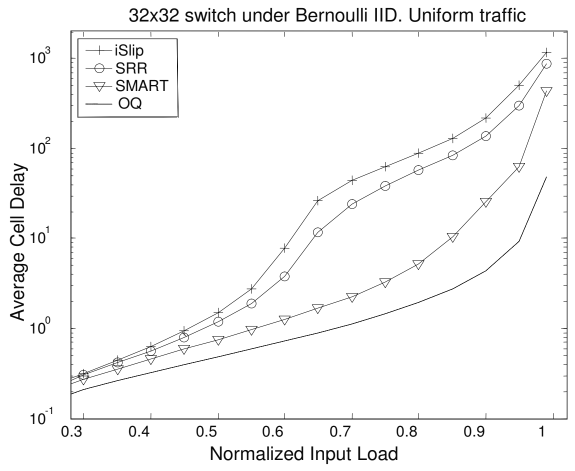

Fig. 3. Average delay performance under Uniform traffic.

As depicted in Figure 3, the SMART algorithm outperforms all other algorithms. The superior performance of SMART is credited to the *pre-Match* phase by eliminating passive requests and therefore reducing the number of communication signals. We believe that the information

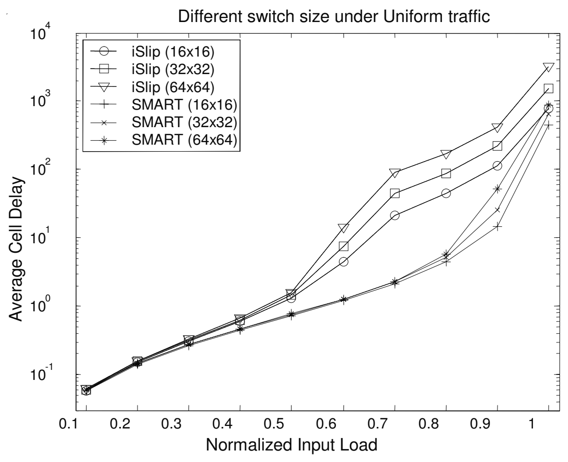

Fig. 4. Different switch sizes under Uniform traffic.

about the newly arriving cells does not contribute under this traffic model. The reason is that under uniform traffic, cells arrival is evenly distributed. Figure 4 shows a comparison between SMART and iSlip under bernoulli uniform traffic and different switch sizes. Again SMART exhibits better delay performance and, better yet, the delay is almost independent of the switch size for loads below 85%.

The bursty traffic is very similar to real life Internet traffic, because cells tend to arrive in trains (or bursts). As depicted in Figure 5, our proposed algorithm has a much better average delay when compared to others schemes. Its

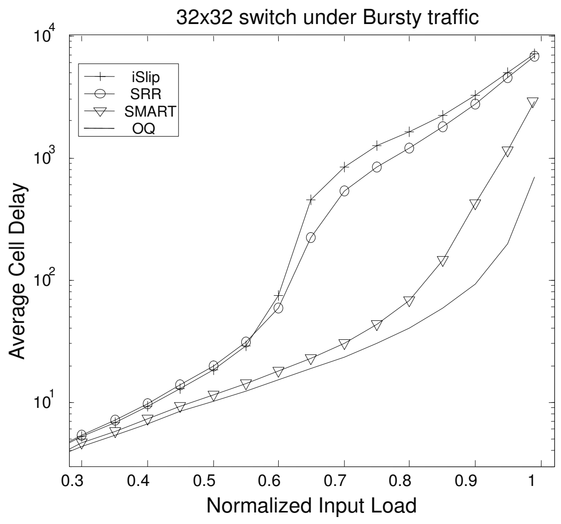

Fig. 5. Average delay performance under Bursty uniform traffic.

performance is quite close to the optimal performance of the OQ switch. While SMART has good performance under bursty traffic, it shows little resistance to the burstiness effect. As depicted in Figure 6, the increase of the average delay of SMART is proportional to the burst length growth, as with iSlip.

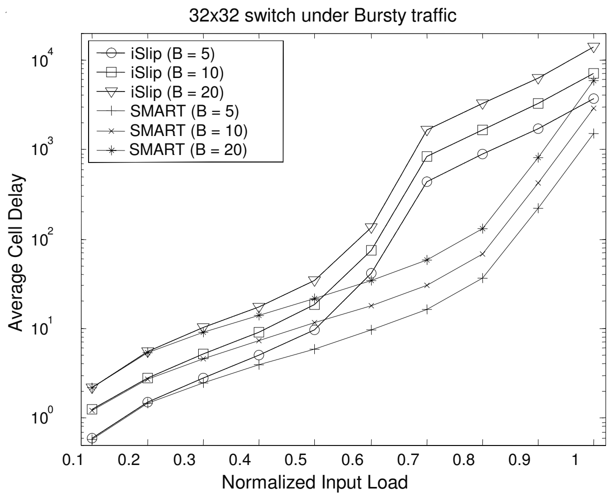

Fig. 6. Average delay performance under different burst lengths, B is the burst length.

In order to test our proposed algorithm under real life traffic, we compare the algorithms under a particular non-uniform traffic. The Diagonal traffic is defined as in the following traffic matrix, for 4×4 switch:

$$\tau(Diagonal) = \rho \begin{pmatrix} \delta & 1-\delta & 0 & 0 \\ 0 & \delta & 1-\delta & 0 \\ 0 & 0 & \delta & 1-\delta \\ 0 & \delta & 1-\delta & 0 \end{pmatrix}$$

This is a very skewed and critical traffic, in the sense that input i has packets only for output i and output $(i+1)(Modulo\ N)$. Unless a MWM algorithm is applied, other schemes will fail to resist this pattern. In the case of SMART and other RR based schedulers, the pointers synchronization problem is unavoidable. Even the information about the cells arrival will not help due to frequent simultaneous arrivals, to different inputs, destined to the same output. Despite all this, as depicted in Figure 7 SMART still has better performance when compared to iSlip and SRR under this traffic model.

V. CONCLUSIONS

Input VOQ switches, using round-robin schedulers, have reputably gained the same high level of high interest in both industrial and academic communities. When RR based schedulers are used, the VOQ switches benefit from the high value of simple hardware requirements and can afford improved levels of fairness. In this paper we analyzed the properties of the RR based schedulers. In particular, we

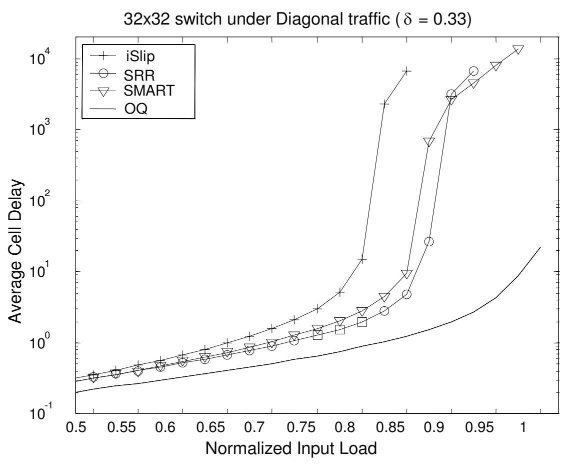

Fig. 7. Average delay performance under Diagonal traffic.

introduced a simple scheduling algorithm called SMART. It is a combination of the simple *pre-Match* (pM) step, the original SRR pointer updating scheme coupled with the extra intelligence of giving priority to newly arriving traffic. SMART was shown to exhibit far better performance than its competitors while being readily implementable in hardware.

REFERENCES

[1] S.T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 06, pp. 1030–1039, Dec. 1999.

[2] W. J. Dally, P. Carvey, and L. Dennison, "The Avici Terabit Switch/Router," *Proceedings of Hot Interconnects 6*, pp. 41–49, Aug. 1998.

[3] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Trans. on Commun.*, vol. 35, no. 09, pp. 1337–1356, Dec. 1987.

[4] N. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*, Ph.D. thesis, University of California at Berkeley, May 1995.

[5] H.C. Chi and Y. Tamir, "Symmetric Crossbar Arbiters for VLSI Communication Switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 04, no. 01, pp. 13–27, Jan. 1993.

[6] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE Communications Magazine*, vol. 36, no. 09, pp. 144–151, May 1998.

[7] A. Mekkittikul, *Scheduling Non-Uniform Traffic In High Speed Packet Switches and Routers*, Ph.D. thesis, Stanford University, Nov 1998.

[8] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks," *ACM Transactions on Computer Systems*, pp. 319–352, Nov. 1993.

[9] N. McKeown, "iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE Trans. On Networking*, vol. 07, no. 02, pp. 188–201, Apr. 1999.

[10] J. Chao, "Saturn: a terabit packet switch using dual round robin," *IEEE Communications Magazine*, vol. 38, no. 12, pp. 78–84, Dec. 2000.

[11] D. N. Serpanos and P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Input Queues," *IEEE INFOCOM*, March 2000.

[12] Y. Jiang and M. Hamdi, "A fully desyncronized Round-Robin Matching Scheduler for a VOQ Packet Switch Architecture," *IEEE Workshop on High Performance Switching and Routing*, pp. 407–411, 2001.