This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Selective Request Round-Robin Scheduling for VOQ Packet Switch Architecture

Dong Lin, Ying Jiang and Mounir Hamdi
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{ldcse, csjyx, hamdi}@cse.ust.hk

***Abstract*—Virtual Output Queuing (VOQ) is widely used by input-queued (IQ) packet switches to eliminate the head-of-line (HoL) blocking problem. A lot of research has been devoted to design iterative arbitration algorithms to maximize the throughput of this architecture. Nevertheless, these approaches require either a high computation complexity or large contention resolution times for high-speed switches. We investigate in this paper the performance of various such algorithms and based on the analysis of pointer desynchronization effect, we propose a new algorithm approximating maximum size matching (MSM) called *Selective Request Round Robin* (SRRR) which performs extremely well under various traffic models and is easy to implement in hardware.**

I. Introduction

The traditional output queuing (OQ) switch structure is appreciated for its optimal performance. OQ switches can always achieve 100% throughput. However, the high internal speed required makes it impractical. In contrast, input queuing (IQ) switches are designed to operate with a switching fabric running at an internal rate equal to the external link speed. Unfortunately, when using a first-in-first-out (FIFO) queuing discipline at the input queues, due to head-of-the-line (HoL) blocking problem, they only provide a maximum throughput of 58.6% [1] under uniform traffic.

An architecture called virtual output queuing (VOQ) [2] is proposed to solve the HoL problem. Rather than maintaining a single FIFO queue for all packets, each input maintains a separate queue for the packets directed to different outputs. Using this architecture, its performance essentially depends on its scheduling (arbitration) algorithm. A good algorithm should achieve high performance, work fast and be simple to implement in hardware.

Many scheduling algorithms have appeared in the literature. They can be classified into two types: approximating maximum size matching (MSM) and approximating maximum weight matching (MWM). Although the MWM algorithms have been proved to achieve 100% throughput under any traffic [4][5], they are too complex to be implemented and have a high running time ($O(N^3 logN)$). A class of matching algorithms based on randomized selection has been proposed to achieve high throughput [14][15]. These randomized matching algorithms require computations and comparisons of the sum of the weight for each matching VOQ-output pair in every time slot, thus the computation complexity is still high. In [16][17], it has been shown that holding the matches of busy flows for longer time is beneficial to achieve higher switching performance than using a complex matching policy. Meanwhile, the performance of cell and packet based switching modes have been shown with some performance differences [18]-[21].

Despite great achievements and advances, traditional single-stage cell-based MSM algorithms still dominate the industrial market because of their simplicity and ease of hardware implementation. Therefore, in this paper, we concentrate on a group of more practical algorithms, including PIM [2], RRM [6], iSlip [6], iLRU [7], DRRM [8], FIRM [9], EDRRM [13] etc., and evaluate their performances.

In this paper, we propose a new algorithm approximating MSM called SRRR (Selective Request Round Robin). It performs much better than the algorithms mentioned above, is simple to implement and also works very fast.

The rest of the paper is organized as follows. In Section II, we introduce some background knowledge of iterative algorithms that approximate MSM. In Section III, we present the formal specifications of our algorithm. The simulation results are demonstrated in Section IV. Section V gives a possible hardware design of our algorithm. Finally, Section VI concludes the paper.

II. Related Work

A. Maximum Size Matching

The scheduling problem in an N x N switch is modeled using a bipartite graph (See Figure 1), where each part contains N nodes, and one part corresponds to the input ports, while the second part to the output ports. The requests from input queues to the corresponding output ports are represented as edges, creating a bipartite graph.

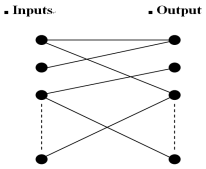

Figure 1. Bipartite Graph

The maximum size matching for a bipartite graph can be found by solving an equivalent network flow problem [10] and the algorithm is called *maxsize* here. The most efficient algorithm converges in $O(n^{5/2})$ time [11]. Although it is guaranteed to find a maximum match, it is too complex to implement in hardware and takes too long time. We will introduce some practical algorithms next.

This research work is partially supported by the following HKUST grant RPC07/08.EG09.

TABLE I. POINTER UPDATING SCHEMES

	Input		Output		
	No grant	**Granted**	**No request**	**Grant accepted**	**Grant not accepted**
RRM	**unchanged**	**one location beyond the accepted one**	**unchanged**	**one location beyond the granted one**	**one location beyond the granted one**
iSlip					**unchanged**
FIRM					**the granted one**

B. A Group of Practical Algorithms

There exists a group of algorithms that are easy to implement in hardware. They only take into consideration whether each VOQ is occupied or not, to make the scheduling decision and try to approximate MSM. The calculation of the matching is performed in an iterative fashion, where each iteration augments the matching calculated so far. In each iteration, the inputs send requests to the outputs, then each output selects one request independently and issues a grant to it, at last each input chooses one grant to accept. In each of the following iterations, only the unmatched inputs and outputs will be considered.

PIM (Parallel Iterative Matching) [2], RRM (Basic Round-Robin Matching) [6] and iLRU (Iterative Least Recently Used) [7] do not provide good performance. The highest throughput under uniform traffic is no more than 65%. Please refer to [6][7] for simulation results. However, it is still meaningful to mention them, because, for example, the idea of RRM was further developed to iSlip, which is currently being used by Cisco products.

The main characteristic of iSlip is it simplicity: it is readily implemented in hardware and can operate at high speed. The performance is also good. For uniform i.i.d. Bernoulli arrivals, iSlip is stable for any admissible load. It means that 100% throughput can be achieved. iSlip works in this way:

Step 1. Request. Each input sends a request to every output for which it has a queued packet.

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 (module *N*) to one location beyond the granted input if and only if the grant is accepted in Step 3.

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 to the highest priority element of the round-robin schedule is incremented (module N) to one location beyond the accepted one.

FIRM (FCFS In Round Robin Matching) was proposed in [9]. The algorithm is almost the same as iSlip. We list in Table I, the difference between RRM, iSlip and FIRM in updating their pointers, which is the main difference among them. The updating scheme plays an important role in improving the performance.

DRRM (Dual Round Robin Matching) scheme [8] is a little bit different from the conventional model we conclude from the previous iterative algorithms: request-grant-accept. It is rather a 2-step algorithm: request and grant. The secret of removing one step is that in the request stage, each input only makes one request instead of sending out all the requests. The inputs choose one among the non-empty VOQs and the outputs choose one among the requests received, using the same scheme of iSlip. By symmetry, we expect the performance will be quite similar to iSlip under uniform traffic. The advantage of DRRM is that it has less control information transmission time.

EDRRM (Exhaustive Dual Round Robin Matching) scheme [13] performs service exhaustion in an achieved match by keeping the match between a queue and an output port until the occupancy of the matched queue is exhausted. This scheme has shown a throughput higher than iSlip and DRRM under non-uniform traffic patterns at the cost of reduced performance under uniform traffic.

III. THE SRRR ALGORITHM

We propose a new algorithm Selective Request Round Robin (SRRR). We will first use an example to explain the idea of this algorithm and then formally specify it.

A. Idea of the Algorithm

This example is generated from the simulation of FIRM for an 8x8 switch under uniform i.i.d. Bernoulli traffic.

Table II shows the pointers to highest priority elements for inputs and outputs at the beginning of a time slot. Table III shows the length of VOQs at the beginning of this time slot. If FIRM is used, the grants sent by the outputs are listed as follows.

Output	1	2	3	4	5	6	7	8
Grant	6	6	4	3	4	1	8	8

Only 5 inputs get granted, so only 5 packets can be sent in this slot. Why is this? The pointers at the outputs are in fact fully desynchronized, such that they do not match with the traffic very well. For example, the highest priority input of output 2 is 5, but there is no traffic requirement from input 5 to output 2, and according to the "choosing the next" scheme, output 2 chooses input 6 instead, which is also selected by output 1. The same thing happens for output 5 and 8. We hope somehow, to prevent output 2 from choosing input 6, as output 1 seems to have more right to choose input 6 as that is its highest priority one. Output 2 fails to choose its highest priority input, but it is better not to "grab" others' highest priority ones.

So the idea is that instead of sending out all the requests, the inputs will choose appropriately part of the requests to send, preventing the outputs conflict if they fail to choose their highest priority inputs. Then how to choose the requests? In the example above, if input 6 knows that itself is the highest priority input of output 1, and it happens that there is a

TABLE II. POINTERS TO HIGHEST PRIORITY ELEMENTS

Output	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Pointer	**6**	**5**	**4**	**3**	**2**	**1**	**8**	**7**
Input	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Pointer	**6**	**5**	**4**	**3**	**2**	**8**	**1**	**7**

TABLE III. LENGTH OF VOQS

Output / Input	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**4**	**0**	**1**	**3**	**5**	**5**	**4**	**1**
2	**3**	**1**	**6**	**4**	**0**	**2**	**2**	**5**
3	**1**	**1**	**6**	**4**	**0**	**11**	**2**	**3**
4	**1**	**6**	**12**	**2**	**1**	**5**	**1**	**6**
5	**1**	**0**	**2**	**7**	**8**	**1**	**3**	**0**
6	**17**	**4**	**5**	**0**	**6**	**1**	**0**	**0**
7	**5**	**2**	**5**	**1**	**3**	**2**	**1**	**0**
8	**8**	**1**	**4**	**9**	**3**	**2**	**2**	**2**

TABLE IV. FILTERED REQUESTS

Output / Input	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1						√		
2	√	√	√	√		√	√	√
3				√				
4			√					
5	√		√	√	√	√	√	
6	√							
7	√	√	√	√	√	√	√	
8							√	

TABLE V. POINTER UPDATING SCHEME OF SRRR

	Input		**Output**		
	No grant	**Granted**	**No request**	**Grant accepted**	**Grant not accepted**
SRRR	**unchanged**	**one location beyond the accepted one**	**incremented by 1**	**one location beyond the granted one**	**the granted one**

packet from input 6 to output 1, it makes sure that output 1 will send a grant to it. So, input 6 just reserves itself for output 1 by only sending this request, thus preventing other outputs sending grants to it.

In Table IV, we list the requests after filtering (the detailed algorithm will be described later). A tick means there is a request. Now let us see how the outputs send grants now:

Output	1	2	3	4	5	6	7	8
Grant	6	7	4	3	5	1	8	2

Each output is granting a different input, so the maximum size matching is made. From this example, we can see that by filtering out some requests which are unlikely to be accepted even if they are granted, we can improve the performance.

B. Algorithm Specification

SRRR is composed of 4 steps: pointer transmission, request, grant and accept.

Step 1: Pointer Transmission.

Each output sends a signal to its highest priority input. In the previous example, output 1 sends a signal to input 6, output 2 sends a signal to input 5, etc. After this step, each input knows which outputs having their highest priorities on it. An input may receive several signals from different outputs as the pointers are not always desynchronized.

Step 2: Request.

For each input, it checks for each VOQ where a signal from the corresponding output is received in step 1.

1. For each such VOQ, if there is any packet waiting, a request will be made for the corresponding output.

2. If for all such VOQs, no packet is waiting, the input will send all the requests.

In the previous example, input 2 gets a signal from output 5, but there is no packet from input 2 to output 5 waiting to be served, so input 2 send all the requests (to output 1, 2, 3, 4, 6, 7 and 8). Input 6 gets a signal from output 1 and there is a packet from input 6 to output 1, then input 6 only sends the request to output 1.

Step 3: Grant.

It works the same as iSlip or FIRM. Pay attention that they are not the same substantially as the sets they choose from are different.

Step 4: Accept.

It is also the same as that of iSlip or FIRM.

The last important thing is how to update the pointers. The input side will be the same as iSlip or FIRM. For the output side, if the grant is not accepted, it will be updated to the granted one like FIRM, as it provides better desynchronization. What is different is that when there is no request received, no matter in iSlip or FIRM, the pointer remains unchanged. In our case, it will be incremented by one. The reason is that not receiving requests doesn't mean there is no traffic actually; doing so helps desynchronizing the pointers under heavy traffic loads. The updating scheme is summarized in Table V.

IV. SIMULATION RESULTS

The simulation results are gathered from a 32x32 switch. Delay is only the period of time a packet spends waiting in a VOQ before being scheduled. Each point in the figures runs for 200,000 time slots (packet time), and the statistics are gathered from the 20,000th time slot. Average delay is got from all the packets outputted during this period of time. Relative average delay is average delay divided by that of an output-queued switch. By normalized load, we mean the percentage of time slots which have packets coming in, averaged over all inputs.

We evaluate the different algorithms under four traffic models: uniform, bursty, hotspot and cross-shaped. For uniform traffic, the packets are Bernoulli arrivals, i.i.d., with destinations uniformly distributed over all outputs. For bursty traffic, busy and idle periods appear alternatively; in a busy period, there is a packet arriving in each time slot; in an idle period, there is no packet arriving in any time slot. The average loads for the inputs are the same, and the destinations are uniformly distributed burst by burst over all outputs. The traffic matrix of hotspot traffic is like (for a 4x4 switch):

$$\begin{pmatrix} 2x & x & x & x \\ 2x & x & x & x \\ 2x & x & x & x \\ 2x & x & x & x \end{pmatrix}$$

If output 0 is the "hotspot" and each flow is Bernoulli, then the traffic matrix of cross-shaped traffic is like (for a 4x4 switch):

$$\begin{pmatrix} 0 & 0 & x & 0 \\ x & x & x & x \\ 0 & 0 & x & 0 \\ 0 & 0 & x & 0 \end{pmatrix}$$

In our simulation, we only consider admissible traffic, which means that no input or output is overloaded. The following figures show the results of one iteration.

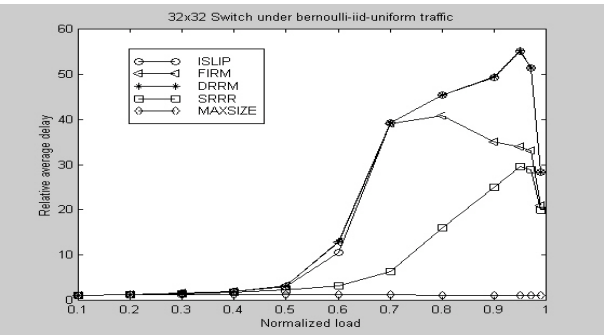

FIGURE 2. RELATIVE AVERAGE DELAY UNDER UNIFORM TRAFFIC

Figure 2 shows the results under uniform traffic. We can see that the performance of iSlip and DRRM is very close. FIRM has better performance than them under high load (>0.7). SRRR has lower average delay than all of the three algorithms for any load, especially when 0.6<load<0.9.

The results under bursty traffic with burst size 16 are shown in Figure 3. FIRM shows no advantage over iSlip and DRRM, while SRRR is still obviously better than the others.

Under hotspot traffic (Figure 4), FIRM and iSlip shows similar performance, with FIRM a little worse. SRRR is still much better than these two algorithms. However, DRRM shows especially good result. It is because the unbalance happens on the outputs, with inputs making decisions of what matches to make; more matches can be made each time. We can imagine that if the hotspot is on an input, the performance of DRRM will not be good.

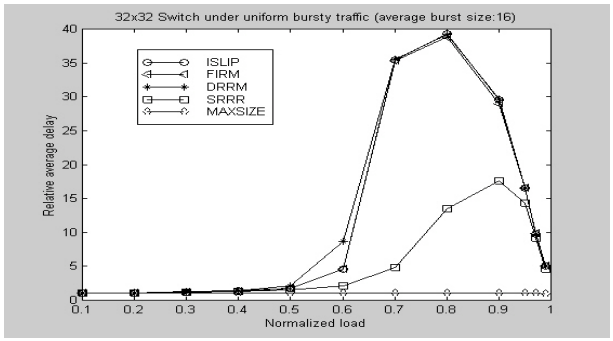

FIGURE 3. RELATIVE AVERAGE DELAY UNDER BUSTY TRAFFIC

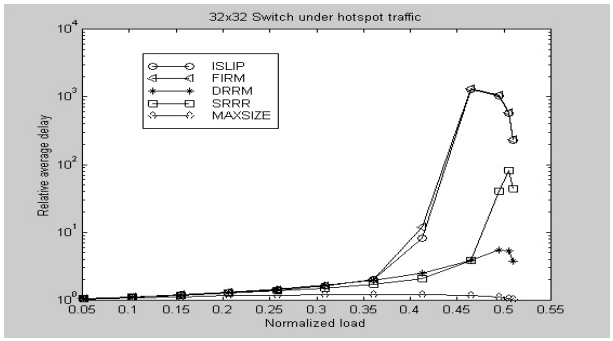

FIGURE 4. RELATIVE AVERAGE DELAY UNDER HOTSPOT TRAFFIC

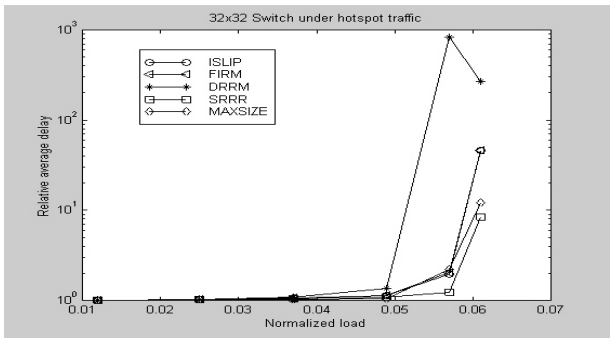

FIGURE 5. RELATIVE AVERAGE DELAY UNDER CROSS-SHAPED TRAFFIC

Under cross-shaped traffic (Figure 5), SRRR is much better than the other algorithms, even maximum size matching (MSM). The performance of DRRM is especially bad under higher load. This is an example to show MSM's instability under non-uniform traffic. Even it can achieve the performance of an OQ switch under uniform traffic, the algorithm itself is too complex for hardware implementation.

From the above results, we can see that our new algorithm SRRR is much better than the other algorithms we have considered no matter in what kind of traffic model we use. In

fact, the traffic models we have considered cover most cases, even some extreme cases. From our simulation results, for more iterations (2-4), SRRR is still better than FIRM, iSlip and DRRM.

V. Implementation of SRRR

We have shown that SRRR has a very good performance. In this section, we will show it is also easy to be implemented.

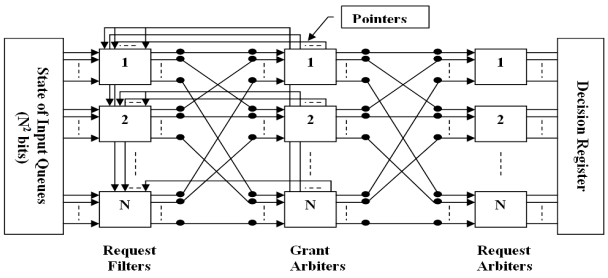

Figure 6. Implementation of SRRR

Figure 6 shows a possible implementation of SRRR. We just add one stage to the iSlip implementation. In this new stage, there are *N* request filters. Each one is responsible for selecting which VOQs to send requests for an input. There are two groups of inputs for each filter: one group is the state of input queues; another group is the signals sent from grant arbiters. Each grant arbiter has maintained a register of *N* bits, with the value of its highest priority input. Here we show a grant arbiter for an 8x8 switch. The second bit of the register is set 1, with others 0, meaning the highest priority input is 2. The 8 bits will be inputs for the 8 filters respectively.

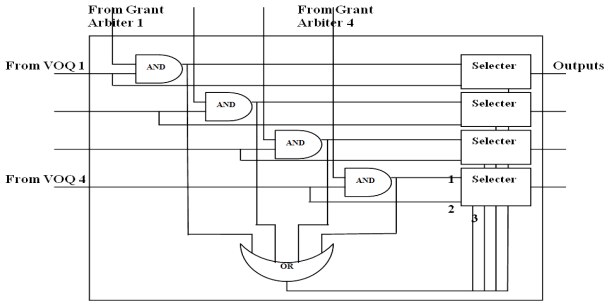

Figure 7. Design of Request Filter

Figure 7 shows the design for each request filter for a 4x4 switch. For each selector, if input 3 is 1, the output will be input 1; if input 3 is 0, the output will be input 2. The time complexity of a filter is only *O(logN)* (because of leveled OR gates), much less than that of an arbiter, which is *O(N),* if ripple design is used. So this new stage only adds little scheduling time to the original iSlip design. For the design of grant and request arbiters, please refer to [6][12].

VI. Conclusion

In this paper, a practical scheduling algorithm for VOQ structure – Selective Request Round Robin (SRRR) is introduced. It achieves much better performance than other practical algorithms under variant traffic models, without adding much complexity to its hardware implementation. SRRR meets the criterion of a good scheduling algorithm: good performance, fast and simple to implement.

References

[1] M.J. Karol, M. G. Hluchyj, and S.P. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," IEEE Transactions on Communications, 35:1347-56, 1987.

[2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks," ACM Trans. Comput. Syst., pp. 319-52, Nov. 1993.

[3] M. A. Marsan, A. Bianco, E. Leonardi, and L. Milia, "RPA: A Flexible Scheduling Algorithm for Input Buffered Switches," IEEE Transactions on Communications, 47: 1921-33, Dec. 1999.

[4] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," IEEE Transactions on Communications, 47: 1260-67, Aug. 1999.

[5] A. Mekkittikul, and N. Mckeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," IEEE INFOCOM 98, San Francisco, April 1998.

[6] N. McKeown, "Scheduling Cells in an Input-Queued Switch," PhD thesis, University of California at Berkeley, May 1995.

[7] N. Mckeown, and T. E. Anderson, "A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches," Computer Networks and ISDN systems, vol.30, pp. 2309-2326, 1997.

[8] H. J. Chao, and J.-S. Park, "Centralized Contention Resolution Schemes for A Large-Capacity Optical ATM Switch," Proc. IEEE ATM Workshop, Fairfax, VA, May 1998.

[9] D. N. Serpanos, and P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-speed ATM Switches with Multiple Input Queues," PROC. IEEE INFOCOM 2000, pp. 548-555. 2000.

[10] R. E. Tarjan, "Data Structures and Network Algorithms," Society for Industrial and Applied Mathematics, Pennsylvania, Nov. 1983.

[11] J. E. Hopcroft, R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," Society for Industrial and Applied Mathematics Journal of Computation, vol.2 pp.225-31, 1973.

[12] N. McKeown, M. Izzard, A. Mekkittikul, and M. Horowitz, "The Tiny Tera: A Small High-Bandwidth Packet Switch Core," IEEE Micro Magazine, vol.17, No. 1, pp. 26-33, Jan-Feb, 1997.

[13] Y. Li, S. Panwar, H.J. Chao, "The Dual Round-robin Matching Switch with Exhaustive Service," IEEE HPSR 2002, pp. 58-63, 2002.

[14] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches", IEEE INFOCOM 98, pp. 533-539, 1998.

[15] P. Giaccone, B. Prabhakar, D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches", IEEE J. Sel. Area Commun. 21 (2003), pp. 546-559, 2003.

[16] R. Rojas-Cessa and C-B. Lin, "Captured-Frame Eligibility and Roundrobin Matching for Input-queue Packet Switches," IEEE Commun. Letters, vol.8, issue 9, pp. 585-587, Sep. 2004.

[17] R. Rojas-Cessa and C-B. Lin, "Captured-frame matching schemes for scalable input-queued packet switches," Computer Communications, May 2007.

[18] H. Kim, J. Son, K. Kim, "A packet-based scheduling algorithm for highspeed switches," Proc. IEEE TENCON, vol. 1, pp. 117-121, August 2001.

[19] C. Hu, X. Chen, W. Li, and B. Liu, "Fixed-length switching vs. variablelength switching in input-queued IP switches," Proc. IEEE Workshop on IP Operations and Management, pp. 117-122, October 2004.

[20] Y. Ganjali, A. Keshavarzian, D. Shah, "Cell Switching Versus Packet Switching in Input-Queued Switches," IEEE/ACM Trans. on Network., Vol. 13, Issue 4, pp. 782-789, August 2005.

[21] Nabeel Al-saber, Saurab Oberoi, Roberto Rojas-Cessa and Sotirios G. Ziavras, "Concatenating Packets for Variable-Length Input-Queue Packet Switches with Cell-Based and Packet-Based Scheduling," Proc. IEEE Sarnoff Symposium, Princeton, NJ, April 28-30, 2008.