Menoufia J. of Electronic Engineering Research (MJEER), Vol. 29, No. 2, Jul. 2020

The Single Bit Request Grant (SBRG) Scheduling Algorithm for Input-Queued ATM Switches

Mervat Said
Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University, Egypt.
mervat.arafa@el-eng.menofia.edu.eg

Zeiad El-Saghir
Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University, Egypt.
ziad.abdoun@el-eng.menofia.edu.eg

Nawal EL-Fishawy
Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University, Egypt.
nelfishawy@hotmail.com

Abstract— **In this paper, we propose an efficient single-iteration single-bit request scheduling algorithm for input-queued ATM switches that based on a new arbitration technique called the Single Bit Request Grant (SBRG) algorithm. The operation of the SBRG depends on the concept known as "the preferred input-output pairs first", and the arbitration requests starts from switch output ports to reduce the number of issued request signals. Compared to other single-iteration algorithms, simulation results show that the SBRG maximizes the match size and improves the switch delay/throughput performance, especially when the load increases. Also, the proposed algorithm reduces the complexity of some of the existing algorithms by decreasing the number of input-output transferred messages, and by the absence of any encoding/decoding mechanism that can be used in some existing algorithms to reduce the size of request signals to one bit.**

Keywords— Input-queued ATM switch, single iteration scheduling algorithm, single-bit request scheduling algorithm.

1. Introduction

Cloud computing architectures provide users with access to applications and services stay in shared server domains through the Internet [1], [2]. IP traffic is transferred in huge amounts between users and server domains, and among too many servers in each domain. Most server domains have storage area networks. These networks facilitate data transfer among disk matrices and the hundreds of servers within different data centers. Networks within data centers have many types, and they must use switches that are characterized by their high speed and low latency to achieve efficient data communication among the elements of these networks [3], [4].

There are many types of switch architectures that are used for data communications in ATM networks. The traditional output queuing switches can achieve a 100% throughput under any traffic load, but it requires high internal speed, which makes it impractical to be built in hardware, especially for large port numbers. In contrast, the switches that based on input queuing operates with an external link speed equals the internal speed of the switch, so it is more proper for fast implementation because it minimizes prerequisite for memory data transmission, where at most one packet is sent or received by an input/output port per a time slot [5], [6]. Iterative algorithms are widely used for the switches that use input queuing [7], [8].

The switch performance is affected significantly by the algorithm of scheduling that can be used for switch arbitration. Many of these algorithms have studied in the literature. They can be classified into two categories: maximal size matching (MSM) and maximal weight matching (MWM). Although the MWM algorithms can achieve a throughput of 100% under any traffic load [9], [10], its implementation is difficult due to its complexity, and also it has a high execution time [O (N^3logN)].

For MSM category of scheduling algorithms, there are a large number of traditional cell-based algorithms [10] - [14] that remains grips the industrial area due to their simplicity and ease of implementation in hardware. Also, another MSM single iteration class of algorithms is introduced in [15], [16], which depend on the scheduling of packets rather than cells. Cell scheduling is simpler to be implemented than packet scheduling.

For a switch that has N inputs and N outputs, the execution of an iterative scheduling algorithm may be repeated up to N iterations to achieve the maximal match size, where each of these N iterations contains three phases: request phase, grant phase and accept phase, as these three phases contribute to increasing the scheduling overhead. The scheduling overhead increases also by increasing the iterations number, which is almost unachievable in high-speed switches.

With a single-iteration algorithm, no acknowledge messages exist in the accept phase. Despite the improvements that have occurred, existing single-iteration algorithms need further improvements. So, a new category of single-iteration algorithms that depend on the idea known as "Highest Rank First" (HRF) and used for the switches with input queues are proposed in [4].

A disadvantage of this new category is the need to an encoding/decoding hardware to produce single-bit requests, which in turn increases the switch complexity.

In this paper, our focus is on some of existing efficient algorithms that depend on the pointer updating schema, including PIM [7], iSLIP [11], SSRR [12], SRRR [14], and CHRF [4] algorithms. We evaluate their performances and compare them to the proposed new algorithm called the "Single Bit Request Grant" (SBRG) algorithm. The proposed SBRG algorithm falls within the category known as "single iteration with single bit request scheduling algorithms". Compared to other single-iteration algorithms, simulation results clearly show that the SBRG algorithm maximizes the match size and improves the switch delay/throughput performance, especially when the load increases. Also, the proposed algorithm reduces the complexity of some existing single-iteration single-bit request algorithms by decreasing the number of arbitration messages that transferred between inputs and outputs, and by not having to use a special encoding/decoding hardware that used in some algorithms to decrease the size of requests to be with a single bit.

The remaining of this paper as follows. In section II, we provide the related work, which is a background study for some MSM scheduling algorithms with an explanation of the effect of the pointer DE synchronization. We introduce the new SBRG algorithm in Section III. Simulation results and comparisons are introduced in Section IV. Finally, the conclusion is introduced in Section V.

2. Related Work

Over the past three decades, numerous algorithms have been introduced that fall under a category known as the "iterative scheduling algorithms". These algorithms differ from others in the information that is sent during the request phase and in the arbitration mechanisms that are used in the remaining phases: grant and accept.

A group of practical iterative algorithms is widely accepted due to the use of massively parallel processing, that calculates the matching in an iterative fashion, making the scheduling decision to consider if VOQs at input ports are occupied or not.

In general, in each iteration of iterative scheduling algorithms, there are many steps, considering only the unmatched inputs and outputs. STEP 1: If an input has a cell waiting on its VOQs for a specific output, the input sends a notification request signal to this output. STEP 2: at each output port, if more than one request-signal arrived from inputs, it selects one of these requests, then this output tells the input if its request is granted or not. STEP 3: if more than one notification grant-signal are received by the input, it accepts one of them.

Parallel Iterative Matching (PIM) [7] and Round Robin Matching (RRM) [13] do not work well. Both achieves a throughput does not exceed 65%. PIM is based on the random selection at the arbiters of input or output ports, which requires more calculations to reach to the maximum matching, so it is difficult to be implemented. iSLIP (iterative round robin algorithm with SLIP) [11] is a modified version of RRM. It provides a 100% throughput for a uniform traffic load and it is simple enough to be implemented in hardware. iSLIP operates in the same manner as RRM, except that updating the grant pointers occurs only if the grant is accepted.

In iSLIP, The RRM step of Grant is as follows: Step 2: Grant. If a request is received by an output, the output chooses the next one using Round-robin scheduling, starting from the element which has the highest priority. The input is notified by the output whether its request is granted or not. If the grant is accepted in Step 3, then the pointer to the element with the highest priority is incremented (modulo N) to one location next to the granted input. This little change leads to the following properties of one-iteration iSLIP algorithm: Property 1: Lowest priority is given to the most recently made connection. Property 2: No starvation for any connection. The main drawback of iSLIP is the need to synchronize arbiter pointers, so it is impractical when the load increases. Also, iSLIP requires at least 4 iterations to reach to the maximum matching, which leads to increasing the delay time.

SSRR is another scheduling algorithm for the switches with VOQs [4], [5]. In SSRR, all input pointers are initially set to zeros, and all output pointers are initialized with input port addresses without duplication. This leads to more fairness when choosing inputs, leading to improved speed of arbiters and reduced execution time. The steps in SSRR iteration are as follows: Step 1. Each input sends a request to each output that has a cell at the queue of this input.

Step 2. If an output receives more than one request, it accepts the one that appears very next in the round-robin schedule, beginning from the highest priority element. Each input is notified by the output whether its request was granted or not. The pointer is incremented to one location beyond the granted one whether the grant is accepted or not.

Step 3. If an input receives more than one grant, it accepts the one that appears very next in the priority table. The pointer is then incremented to one location beyond the accepted location. The pointer unchanged if no grant is received. The main drawback of SSRR is that it requires O (log N) time complexity.

The Selective Request Round Robin (SRRR) is another Scheduling algorithm proposed in [14]. The SRRR iteration has 4 steps as follows: Step1. Pointer transmission: a request is sent from every output to its input that has the highest priority, where the preferred input is selected in the same manner as iSLIP and IRRM. Several signals may be sent to an input from different outputs. Step2. Each input checks the VOQ of the output that sent a signal to this input in step 1. If there is a waiting packet for each such VOQ, a request will be sent for the corresponding output. If no packet is waiting for all such VOQs, the input will send requests to all outputs. Step3. Grant. SRRR works in the same manner as iSLIP.

Step4: Accept. SRRR works in the same manner as iSLIP.

The main difference between the above four algorithms is in the pointer updating schemas at the arbiters of the inputs and the outputs as shown in Table 1. The significant impact of pointer updating schemas on the performance of a scheduling algorithm is obvious from the table. We also notice that the studied algorithms in Table 1 are similar at the input side, but they differ at the output side.

3. Our Proposed

3.1 SBRG Switch Fabric

A general description of a switch with VOQs with distributed output arbiters is shown in Fig 1 [16]. This architecture describes an N×N single-stage crossbar switch, where N is the number of input/output ports. An incoming traffic (small fixed size cells with 53 bytes' length that is easier to synchronize and schedule) are buffered in separated virtual queues at input ports, where every input has N of VOQs (one VOQ for each output port). There is one arbiter for each output port to keep track of input ports those have packets destined to it and their arrival order. These arbiters can be located on the input ports' side. No memory required for arbitration at output ports. The output arbiters are independent of each other and distributed on output ports. These output arbiters are independent of each other and can be implemented using simple FIFO buffers. All output arbiters can be implemented in hardware on a single chip.

3.2 SBRG Specification

Initially, the pointer of each output arbiter is set to its highest priority output without duplication, and then the steps of SBRG will be as follows:

Step 1: Request selection. Each output sends a signal to the input that currently has the highest priority. At this point, each input knows which output considers this input as its highest priority one. By this way, no synchronization is required between pointers because each input receives only one signal from the output that considers this input as its highest priority input.

Step 2: Request. Each input checks it's VOQs, and if there is a cell waiting for its preferred output (the output that sent a signal to this input in step 1), the input sends a request bit to this output. Otherwise, the input sends a single-bit request to each output that has a waiting cell in its VOQ at this input. This request bit is binary 1 if a cell is queued to this output or binary 0 if output's VOQ is empty.

Step 3: Grant. At each output port, it checks the incoming requests, and if received from its highest priority input, the output grants it by sending a single bit to this input. Otherwise, the output selects the next input from its FIFO buffer.

Step 4: Accept. If a grant is received by an input, this input accepts the one that appears next in a fixed round-robin schedule starting from the element with the highest priority, then all the pointers are incremented by one regardless of whether there is a grant or not.

Algorithm 1 describes the SBRG pseudo code. It should be noted that:

(a) At input ports, all Request and Accept procedures run in parallel with no messages need to be sent in accept procedures.

(b) At output ports, Grant procedures run in parallel.

Algorithm 1: The SBRG Algorithm.

```
Initialization:
 1:     // M: switch size
 2:     // outp: output port
 3:     // inp: input port
 4: for outp = 0 to M-1 do
 5:         // calculate the highest priority input at time slot ti
 6:     inp = (outp + M - ti modM) modM
 7:         pc [outp] = inp // pc: priority calculation
 8: end for
 9: Procedure Request (inp, outp)
10:       // L[inp]: is the length of VOQ at input inp
11: If pc [inp] ==1 and L[outp] > 0 then
12:     send 1 to the output outp
13:     send zero to other outputs
14: else
15:       // qstate [inp] is the status of VOQ at input inp
16:         send each qstate [inp] to the crossponding output
17: end if
18: end Procedure
19: Procedure Grant (outp, ti)
20:       // Req [inp] is the request from VOQ at input inp
21:     inp = (outp + M - ti modM) modM
22: If Req [inp] == 1 then
23:     send 1 to input inp
24: else
25:       k = get inp at the head of qstate [outp]
26:     send 1 to input = k
27:     send zero other inputs
28: end if
29: end Procedure
30: Procedure Accept (inp, ti)
31:         // Gra [outp] is the grant from output outp to input inp
32:     outp = (inp + ti) modM
33: If Gra [outp] == 1 then
34:     accept the grant from output outp
35: end if
36: end Procedure
```

Table 1. Pointer updating schemas for the studied scheduling algorithms.

Algorithm	**Input**		**Output**		
	No confer (No grant)	***Conferred (Granted)***	***No request***	***Acceptance of grant***	***Non-Acceptance of grant***
SSRR	Pointer remains at the same location without any change.	Pointer update to one location following the accepted one.	Pointer remains at the same location without any change	Pointer update to one place beyond the granted one	
RRM				Pointer update to one place behind the granted one	Pointer update to one place beyond the granted one
iSLIP					Pointer remains at the same location without any change
SRRR			Pointer increme nt by one		Pointer update to the granted one

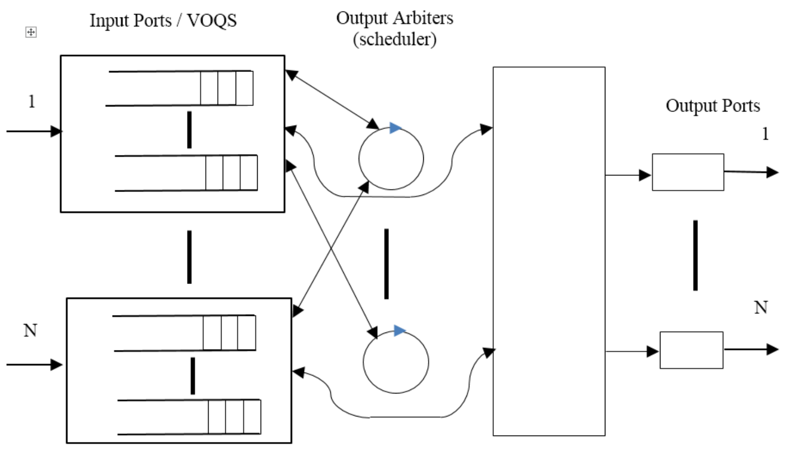

Fig 1. A General Description of an IQ Switch Fabric with distributed output arbiters [16].

Table 2. Pointer updating schemas for SBRG.

Algorithm	**Input**		**Output**		
	No-grant	***Granted***	***No request***	***Grant accepted***	***Grant not accepted***
SBRG	Pointer remains at the same location without any change.	Pointer update to one location following the accepted one.	Increment Pointer by one		

Table 2 shows the pointer updating schema for the SBRG algorithm. Pointer updating is as follows: At each input port: If no grant is received from an output, the Pointer doesn't change. If a grant received, the Pointer is updated to one location following the accepted one. At the output port: If no request is received from an input, the highest priority pointer at the round-robin arbiters updated by 1 to give another input the chance to be served. The output pointer is incremented by one whether the grant is accepted or not.

3.3 SBRG Throughput

The proposed SBRG algorithm achieves a 100% throughput and compared to other single iteration switches that achieve a 100% throughput the SBRG is more distinguished in terms of stability especially when the load increases, as shown in Figure 2. To find the achieved throughput by the SBRG algorithm, let's assume that each input port has M VOQs. let's assume also that all the VOQs have cells need to be transferred to the same destination output port outp. Where the VOQs at inputs try to transfer their cells in certain time slot ti.

Assume that TH is the throughput of input ports VOQs, so TH represents the overall throughput of the switch. Moutp represents the total number of HOL cells destined to an output outp in certain time slot ti. At all the VOQs, the overall number of HOL cells blocked at the input ports in ti is:

$$MB = Moutp - X(Moutp) \qquad (1)$$

where X(Moutp) = min (1, Moutp), is the minimum number of blocked cells. When SBRG sends up to M cells out and tries to transfer these M cells to the preferred output ports at a time slot ti, and at the steady state S[X(Moutp)] = TH, so by substituting in (1) gives

$$TH = S[Moutp] - S[MB] \qquad (2)$$

Let L is the all unblocked VOQs at time slot ti. So

$$L = M - MB \qquad (3)$$

With keeping the flow, and let Y be the possibility of one of the L unblocked VOQs at input ports gets a new cell at a time ti. We found that

$$S[L]Y = TH \qquad (4)$$

From (3) and the compensation in (4), we obtain

$$S[MB] = M - TH/Y \qquad (5)$$

Where the HOL cells number that destines to an output outp in a time ti + 1 is M'outp. Let Voutp is the total number of HOL cells destined to the output outp and arrived at the L VOQs. So

$$M'outp = Moutp - X(Moutp) + Voutp \qquad (6)$$

Using the equation of the mean value in Appendix A of [17] we obtain:

$$S[Moutp] = S[Voutp] + S[Voutp(Voutp - 1)]/2(1 - S[Voutp]) \qquad (7)$$

For large M, and by using a Poisson distributed random variable, Voutp can be simplified.
Using the proof in Appendix A of [5], where at the steady state. When $M \rightarrow \infty$, the overall number of HOL cells at the VOQs with the destination outp at every time slot becomes Poisson.

So, we found that

$$S[Voutp] = S[X(Moutp)] = TH \qquad (8)$$

and

$$S[Voutp(Voutp - 1)] = TH^2 \qquad (9)$$

From (8) and (9) and commute (5) and (7) into (2), and setting Y = 1:

$$TH = 1 + M - \sqrt{(1+M^2)} \qquad (10)$$

When $M \rightarrow \infty$, the throughput is 1. In fact, when M is finite, the equation remain gives a result that is very close to the maximum throughput value.

3.4 SBRG Complexity

SBRG achieves a maximum matching in only one iteration, and all of its operations take the same time to be finished, so it provides fast scheduling. The time complexity of SBRG is O(1), in contrast to iterative algorithms that perform multiple iterations to reach the maximum matching, making them to be too slow to support high line rates.
SBRG needs fewer messages to work than iterative matching algorithms. For each matching, an output arbiter sends only one message as a request and only one grant message. For each input, it can send up to N one-bit request messages depending on the VOQs status (one request for every output having a cell in its VOQ). Over the entire switch fabric, there are a maximum of 2N messages that may be exchanged between the inputs and the outputs. Unlike SBRG, iSLIP needs a number of requests equals N^2, a number of grant notifications equals N^2, and a number of accepts equals N for each iteration. Also, iSLIP needs at least a number of iterations equals log N to achieve the maximum matching. So, iSLIP needs a number of messages equals $(2N^2+N)$ log N. Also, PIM and SSRR algorithms need a number of messages like the case of iSLIP. For SBRG, N selection requests are sent from output to input ports and only N grants are sent from output ports to input ports, so a total of N+N bits are exchanged. In contrast, in iSLIP and other iterative algorithms, a number of bits equal to $2N^2+N$ are required to be exchanged for only one iteration. Also, and compared to the single iteration HRF rank-based algorithm given in [9], HRF time complexity is O(log N). Also, compared to the CHRF as a single iteration rank-based algorithm given in [9], the CHRF

time complexity is O(1), but it increases the complexity of the hardware implementation, which is very high and considered as a main drawback of the CHRF.

4. SIMULATION RESULTS

4.1 Simulation Setup

A performance comparison is done for our SBRG algorithm to some of the best performance and widely used single-bit-single-iteration algorithms, namely CHRF, SSRR, SRRR, PIM-1, and iSLIP-1 algorithms. Simulations are performed and the results are obtained using 16×16, 32×32, and 64×64 switches. The considered performance metrics are the cell delay time and the average throughput under normalized traffic load, where the cell delay is the time that a cell waits in a VOQ until being scheduled to its preferred output, and the traffic load is the cells number per time slot for each input. The used switch fabric is a single-stage crossbar one (the use of crossbar for building the connection between input ports and output ports is more preferred due to the simple design of crossbar and its non-blocking property). An incoming traffic (small fixed size cells 53-bytes each) are buffered in separated virtual queues at input ports. The access rate of the SBRG switch is identical to the access rate of the connection links outside the switch, so switch speedup is not required. Simulation results are obtained using a uniform traffic in one iteration (when cells arrive at the input it destines to each output with equal probability).

4.2 Evaluation Metrics

(i) Throughput: The algorithm should provide a good performance of throughput. Throughput defined as the ratio of the actual number of cells enters the switch successfully to the maximum possible number of cell arrivals during the simulation.
(ii) Delay (average cell delay): The algorithm should provide a good performance of delay, defined as the total time the cell spend inside the buffer until it arrives at its output.
(iii) Fairness: The algorithm must serve all the input ports in a fair manner.
(iv) No Starvation: the algorithms should be starvation-free.
(v) Implementation and time complexity: the hardware implementation of
(vi) the algorithm must be simple.
(vii) Scalability: The algorithm should be scalable if the number of ports increases.

4.3 Results

From throughput simulation results shown in Fig. 2, and under a light traffic load, all the algorithms have almost the same performance until the load value 0.6 is reached. Beyond this point, the performance of PIM algorithm becomes unstable, and the maximum throughput that can be offered using PIM is 0.65. Also, the SSRR performance degrades after the point in which the incoming traffic load with the value 0.7, but it still better than iSLIP under all load conditions.

For SBRG algorithm, simulation results show that its throughput is the best among all the studied algorithms, and it is the more stable and the closest one to the 100% throughput under all load conditions.

Cell-delay simulation results are shown in figures 3-a, 3-b, and 3-c. From the results, and under a light traffic load, it is clear that all the algorithms have almost the same cell-delay time until the load reaches the value 0.6. Beyond this point, the performance varies significantly depending on the algorithm used as follows: beyond a 0.6 load value, the performance of PIM algorithm becomes unstable and the cell-delay time increases significantly.

The same situation is applicable to SRRR algorithm beyond a 0.7 load value. Also, simulation results show that SSRR works much better than iSLIP under all load conditions. For the SBRG and compared to the other studied algorithms, a significant performance gain is achieved as the traffic load increases.

The SBRG distinction is clearly noticed with increasing the load. Table 3 shows the performance gain of the SBRG algorithm compared to the studied algorithms as the traffic load increases. For example, if the applied load is 0.8, the cell delay time for the SBRG for a 64×64 switch compared to the studied switches will be as follows: SBRG = 10.239, CHRF = 28.7, PIM = 10050, SRRR = 2321.987, iSLIP = 177.34, and SSRR = 79.421.

Fig 4 shows the SBRG performance as a function of the switch size. The simulation results show that the SBRG performance has a slight change with the increased size of the switch. For example, if the applied load is 0.8, the cell delay time for the SBRG for a 64×64 switch is 10.239, for a 32×32 switch is 9.9293, and for a 16×16 switch is 8.4239. These results prove that the SBRG is a scalable algorithm.

5. Conclusion

In this paper, an efficient scheduling algorithm for switch architectures with VOQs – Single Bit Request Grant (SBRG) is introduced. SBRG achieves a distinct performance compared to other widely used single iteration algorithms, especially when the load increases. Also, the proposed algorithm reduces the complexity of some existing single-iteration single-bit request algorithms by decreasing the number of input-output transferred messages, and by the absence of a request encoding/decoding hardware which can be used by some algorithms to reduce the size of the request to be a single bit. SBRG meets the criterion of a distinct scheduling algorithm: distinct performance, and fast and simple to be implemented.

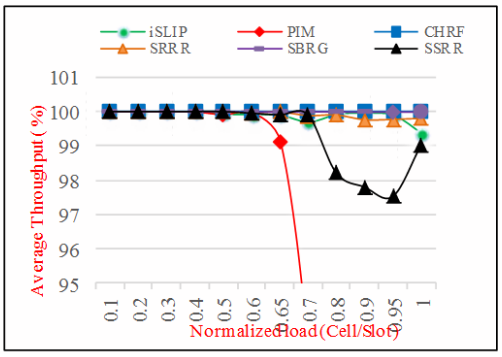

Fig. 2: Average Throughput for a 64×64 switch.

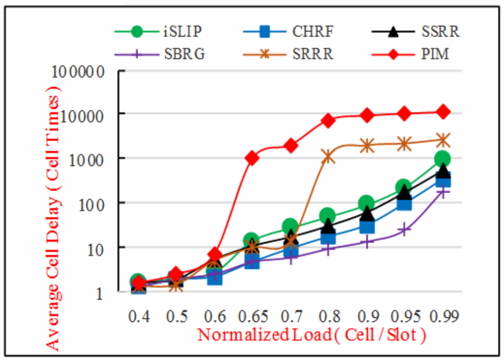

Fig. 3-a: Average cell delay for a 16×16 switch.

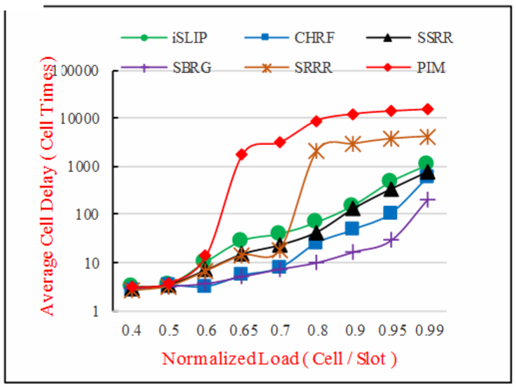

Fig. 3-b: Average cell delay for a 32×32 switch.

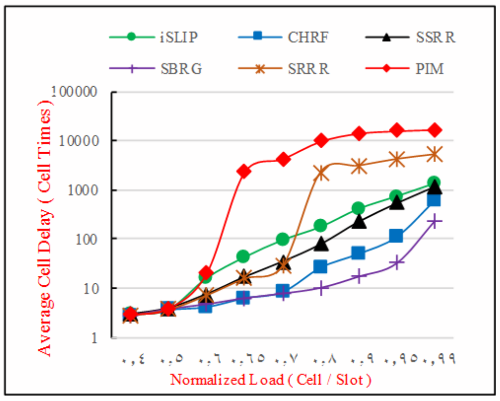

Fig. 3-c: Average cell delay for a 64×64 switch.

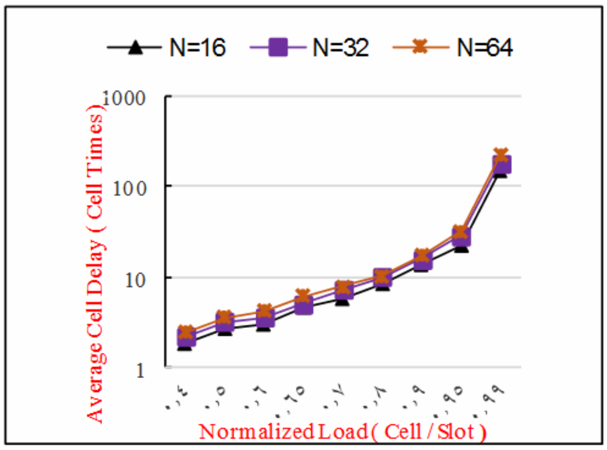

Fig. 4: The performance of SBRG as a function of switch size.

Table 3. Average Cell Delay Comparison under Heavy Load Conditions.

Load	Size	Average Cell Delay					
		SBRG	CHRF	SSRR	SRRR	PIM-1	iSLIP-1
0.7	16*16	5.859	6.52	17.14	14.01	2033.6	27.1214
	32*32	7.245	7.845	22.893	18.23	3160.68	83.31
	64*64	7.8201	8.459	35	28.789	4393.8	95.98
0.8	16*16	8.4239	17.63	30.47	1182.1	7500.57	48.301
	32*32	9.9293	25.7	42.723	2098.01	9050.75	66.53
	64*64	10.239	28.7	79.421	2321.987	10050	177.34
0.9	16*16	13.378	32.1	61.542	1999.4	9324.1	89.757
	32*32	16.678	48.1	134.23	2976.94	12400.3	150.894
	64*64	17.379	50	230	3124.7	14589.6	413.08
0.95	16*16	24.574	98.5	172.92	2197.3	10731.8	222.896
	32*32	28.754	105	340.1	3794	14280	462.7
	64*64	32.564	110	550	4276.45	16388.8	743.1

References

[1] E. Zahavi, I. Keslassy, and A. Kolodny, “Distributed adaptive routing Convergence to non-blocking DCN routing assignments,” IEEE J. Sel. Areas Commun., vol. 32, no. 1, pp. 88-101, Jan. 2014.

[2] Z. Cao and S. S. Panwar, “Efficient buffering and scheduling for a single chip crosspoint-queued switch,” IEEE Trans. Commun., vol. 62, no. 6, pp. 2034-2050, Jun. 2014.

[3] A. Singh et al., “Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network,” ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 4, pp. 183-197, 2015.

[4] BING HU, FUJIE FAN, KWAN L. YEUNG, SUGIH JAMIN, “Highest Rank First: A New Class of Single-Iteration Scheduling Algorithms for Input-Queued Switches” IEEE/ ACM Transactions on Networking, Vol. VOLUME 6, pp.11046-11062, 2018.

[5] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, “Input versus output queueing on a space-division packet switch,” IEEE Transactions on Communications, COM- 35(12), pp.1347–1356, December 1987 .

[6] Tamir, Yuval, and Gregory L. Frazier. “High-performance multi-queue buffers for VLSI communications switches”. IEEE Computer Society Press, Vol. 16. No. 2. 1988 .

[7] Yun, Z., Peng, L. & Zhao, W., RR-LQD:” A novel scheduling algorithm for CICQ switching fabrics”, Proc. 15th Asia-Pacific Conference on Communications (APCC), 2009, pp. 846-849.

[8] X.T. Wang, Y.W. Wang, S.C. Li, P. Li,”A Novel High-Performance Scheduling Algorithm for Crosspoint Buffered Crossbar Switches” International Conference on Computer Information Systems and Industrial Applications (CISIA 2015), pp. 2105-2115.

[9]

[10] Mekkittikul, Adisak, and Nick McKeown. “A practical scheduling algorithm to achieve 100% throughput in input-queued switches.” INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. IEEE, 1998. p. 792-799.

[11] Anderson, Thomas E., et al. “High-speed switch scheduling for local-area networks.” ACM Transactions on Computer Systems (TOCS), Vol. 27. No. 9. ACM, 1992.

[12] N. McKeown, “The iSLIP: A Scheduling Algorithm for Input-Queued Switches,” IEEE Transactions on Networking, Vol 7, No.2, April 1999.

[13] Afridi, Sharjeel et al. “The Quantitative Analysis of Round Robin Matching Scheduling Algorithms for VOQ Packet Switch Architecture.” International Journal of Electronics Communication and Computer Engineering.(2012)

[14] McKeown, Nick. “Scheduling Cells in Input-Queued Cell Switches”. Diss. PhD. Thesis, University of California, Berkeley, 1995 .

[15] D Lin, Y Jiang, M Hamdi, “Selective Request Round-Robin Scheduling for VOQ Packet Switch Architecture”, IEEE International Conference on. IEEE, 2011. p. 1-5.

[16] Jie Xiao and Kwan L. Yeung, “Iterative Multicast Scheduling Algorithm for Input-Queued Switch with Variable Packet Size” IEEE 30th Canadian Conference on. IEEE, 2017. p. 1-4.

[17] Gao, Ya, WeiTao Pan, and Ling Zheng. “Improved analytical model for performance evaluation of crosspoint-queued switch under uniform traffic.” IET Networks, 2017, 6.4: 81-86.

[18] H. Kim and K. Kim, “Performance analysis of the multiple input-queued packet switch with the restricted rule,” IEEE/ ACM Transactions on Networking, Vol. 11, No. 3, pp. 478-487, June 2003 .