Title	D-LQF: An efficient distributed scheduling algorithm for input-queued switches
Author(s)	He, C; Yeung, KL
Citation	The 2011 IEEE International Conference on Communications (ICC 2011), Kyoto, Japan, 5-9 June 2011. In Proceedings of the IEEE ICC, 2011, p. 1-5
Issued Date	2011
URL	http://hdl.handle.net/10722/140253
Rights	IEEE International Conference on Communications. Copyright © IEEE.

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

D-LQF: An Efficient Distributed Scheduling Algorithm for Input-Queued Switches

Chunzhi He and Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC
Email: {czhe, kyeung}@eee.hku.hk

***Abstract*—Iterative scheduling algorithms are attractive in finding a maximal size matching for an input-queued switch. For constructing a large high-speed switch, a distributed multi-chip implementation of an iterative scheduling algorithm should be followed. Since different chips may locate on different switch linecards and linecards can be separated by tens of meters, the propagation delay between chips/linecards is non-negligible. This calls for a pipelined implementation of a single-iteration scheduling algorithm. To this end, a new packet scheduling algorithm called Distributed Longest Queue First (D-LQF) is proposed in this paper. In D-LQF, exhaustive service policy is adopted for reusing the matched input-output pairs in the previous time slot. This helps to maximize the size of match in each time slot. To avoid incorrectly granting an empty VOQ the chance to send a packet, each output keeps track of the lengths of all VOQs destined to it. As compared with other single-iteration scheduling algorithms, simulation results show that our D-LQF provides the best delay-throughput performance.**

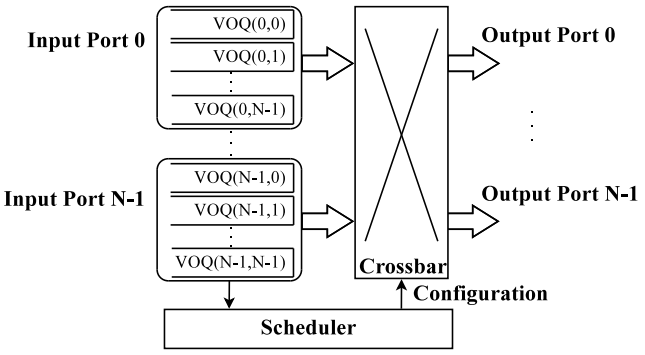

Fig. 1. An input-queued switch with a centralized scheduler. Note that input i and output i are located on the same linecard i.

I. Introduction

Due to the wide use of WDM (Wavelength Division Multiplexing) technology in fiber, the transmission capacity increases sharply, while the processing capacity of current commercial routers increases slowly. This speed mismatch makes the need for building high speed routers urgent [1]. It is well known that output-queued switches can achieve optimal delay-throughput performance. But the associated high speedup requirement on both switch fabric and output port buffers makes output-queued switches difficult to scale.

Accordingly, input-queued switch architecture becomes the preferred choice because no (or little) speedup is required. In an input-queued switch, each input/output port can send/receive at most one packet per time slot. Packet contention occurs and a centralized scheduler is adopted to resolve the contention. To eliminate the phenomenon knowns as head-of-line blocking, virtual output queueing (VOQ) is usually adopted (as shown in Fig. 1), where a dedicated queue is maintained at each input for packets destining to different outputs.

The scheduling problem in an input-queued switch is equivalent to the matching problem in a bipartite graph. Although maximum weight matching and maximum size matching algorithms can be used for obtaining the optimal throughput performance, their high time complexities of $O(N^3)$ and $O(N^{2.5})$ [1], where N is the switch size, are prohibitive for high-speed implementation. To have a faster scheduling algorithm, *maximal* size matching is adopted where backtracking on already matched input-output pairs is not allowed. There are various ways of implementing a maximal size matching algorithm [2]–[5]. Among them, the class of iterative scheduling algorithms (e.g. [2], [3], [5]) is most popular due to their massive use of parallel and distributed operations of inputs and outputs. Each iteration consists of three phases of operations, request, grant and accept (see Section III). A selector is required at each input/output for determining the wining grant/request. To guarantee the maximal size matching, N iterations are required in each slot.

Following the traditional monolithic implementation [6] of iterative scheduling algorithms, all input and output selectors are bound together on the same chip as shown in Fig. 2(a). They can exchange state information and decisions without any communication latency. But this monolithic implementation is not scalable when the switch size is large and the distance between linecards is significant (e.g. tens of meters). This calls for a multi-chip implementation of the scheduler. Recently, several scheduling algorithms [7]–[9] suitable for multi-chip implementation have been proposed. A distinct

This work is supported by Seed Funding Programme for Basic Research 200911159103, The University of Hong Kong.

978-1-61284-233-2/11/$26.00 ©2011 IEEE

feature is that they all adopt a single-iteration operation.

In this paper, a new distributed single-iteration scheduling algorithm called Distributed Longest Queue First (D-LQF) is proposed. In D-LQF, longer VOQ is given has scheduling higher priority. In order to maximize the size of the match in each time slot, efforts are made in keeping/reusing the connected input-output pairs in the previous time slot until the corresponding VOQs are exhausted (or a pre-defined number of cells have been served). To avoid incorrectly granting an empty VOQ the chance to send a packet, each output keeps track of the lengths of all VOQs destined to it. Simulation results show that our D-LQF yields the best delay-throughput performance among all existing distributed algorithms. The rest of the paper is organized as follows. In the next section, recent efforts on designing single-iteration scheduling algorithms are reviewed. Our D-LQF is detailed in Section III. In Section IV, simulation results are presented and we conclude the paper in Section V.

II. Single-Iteration Scheduling Algorithms

By simply limiting the number of iterations used in an iterative algorithm to one, a single-iteration version can be obtained. Notably, iSLIP [3] and iLQF [5] are two classic iterative algorithms. Their single-iteration versions can be implemented using the monolithic approach in Fig. 2(a), and the handshaking between input and output selectors is shown in Fig. 3. We denote the time between sending requests and receiving grants as one round trip-time (RTT). A general requirement is that in each time slot, one RTT is required for scheduling and the remaining time is for packet transmission.

To enhance the performance, some *tailor-made* single-iteration scheduling algorithms are designed from scratch. To the best of our knowledge, there are three tailor-made single-iteration algorithms, SRR (Synchronous Round Robin) [7], π-RGA [8] and SRA (Single Round-robin Arbitration) (and its variant SRA+) [9]. SRR [7] is based on a two-phase operation of request-grant. In slot t, each input i sends a request to its *preferred* output j if VOQ(i, j) is not empty, where $j = (i + t) \, mod \, N$. If VOQ(i, j) is empty, a request corresponding to the current longest VOQ at input i is sent. In the grant phase, an output grants the preferred request. If no preferred request is received, a request will be granted randomly.

Unlike SRR, π-RGA algorithm [8] makes an explicit effort in keeping the match in consecutive time slots "stable" by reusing some matched input-output pairs in the previous time slots. In π-RGA, this is achieved by dividing all requests into *strong* and *weak* based on the active time of individual VOQs. Strong requests are given a higher scheduling priority to form a "stable" base set, whereas weak requests are used to further expand the size of match in each slot.

In SRA [9], each output maintains the occupancy information for all VOQs destined to it and grants one non-empty VOQ in the round-robin order; if an input receives multiple grants, it accepts *all* of them. Then all accepted packets are sent simultaneously. In the worst case, this requires a speedup of N times, and is thus not scalable.

The implementation of a scheduling algorithm can be carried out based on four different levels of distribution [7] as summarized in Fig. 2. Specifically, Fig. 2(a) is the traditional monolithic/centralized implementation, whereas Fig. 2(d) is the fully distributed implementation with all input selectors (ISes) and output selectors (OSes) reside on distinct chips/linecards. In between, we have two partially distributed implementations: Fig. 2(b) assumes both all ISes are collocated on a chip and all OSes are on another, and Fig. 2(c) allows ISes to be located on distinct chips/linecards. There is no doubt that a large-scale high-speed switch prefers the fully distributed implementation in Fig. 2(d). It should be noted that due to the large distance between switch linecards, the RTT between IS and OS can be of multiple time slots. In this case, no matter what speedup is used for packet transmission, the switch will be blocking. To address this RTT problem, pipelined scheduling is required. Notably, SRR readily works in a pipelined fashion, where an input (selector) can send multiple requests (one in each time slot) while waiting for a grant to arrive RTT slots later.

In fact, i-SLIP has also been extended to i-$\triangle$SLIP [10] to address the RTT issue above. Each output is equipped with N counters to keep track of the length of all VOQs destined to it. Both packet arrival signals and arbitration decisions are sent in a pipelined fashion. But i-$\triangle$SLIP requires the centralized implementation in Fig. 2(a) and thus not scalable.

III. D-LQF: Distributed Longest Queue First

A new single-iteration, fully distributed scheduling algorithm called Distributed Longest Queue First (D-LQF) is proposed in this section. Aiming at maximizing the size of the match, the basic version of D-LQF is presented first while assuming the RTT is small. Then two refinements are introduced to deal with the non-negligible RTTs and the starvation problem under some inadmissible traffic patterns.

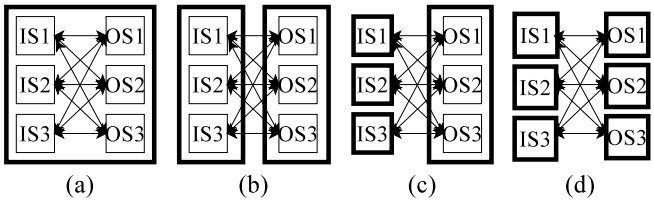

Fig. 2. Four distribution levels. IS = input selector, OS = output selector.

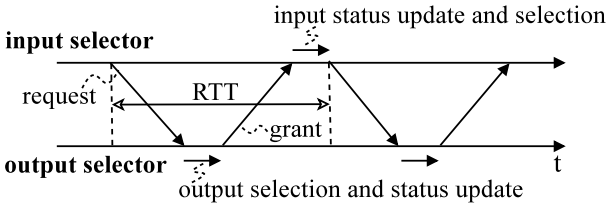

Fig. 3. Round-trip time between input and output selectors.

A. Basic scheme

Reusing the match in the previous slots is a good way to improve the quality of the match in the next time slot. π-RGA [8] makes an explicit effort in reusing the match in the previous time slots via the notion of strong/weak requests; SRR makes an implicit effort via its preferred input/output concept. One extreme in reusing the previous match is that once an input-output pair is matched, all packets belonging to the corresponding VOQ will be served continuously until the VOQ becomes empty. This is known as the *exhaustive service policy* [11] in polling systems. (This policy has been applied to (multiple) iterative scheduling algorithms DRR [4] and i-SLIP, and the resulting algorithms are called EDRRM [12] and E-iSLIP [13].)

In this paper, we apply the exhaustive service policy to LQF scheduling with the following three-phase operation at slot t:

Request: For input i, if a packet of VOQ(i, j) was served in slot $t-1$ and VOQ(i, j) is non-empty, send a request to output j; otherwise send requests for all non-empty VOQs with their respective queue lengths.

Grant: For output j with R being the set of requests received, if R contains request from input which was served by it in slot $t-1$, grant this request; otherwise grant the request with the longest queue length and a tie is broken randomly.

Accept: For input i, accept the grant with the longest queue length. In case of a tie, select a winner randomly.

Unlike scheduling algorithms with multiple iterations, the basic D-LQF scheme above spreads its efforts in achieving the maximal matching over multiple slots. In the request phase, a single request is issued by an (engaged) input if the input was served in the previous time slot and it still has packets from the same flow to send. This single request is guaranteed to succeed, so generating more requests (for other non-empty VOQs) is useless in further enhancing the performance of the engaged input, and even worse, it may allure some outputs to incorrectly grant it. Therefore, in the current slot, the previous matched input-output pairs are reused and new input-output pairs are added. One interesting result is that efforts spent on matching become cumulative over time.

B. Refinement 1: dealing with RTTs

In a fully distributed environment (Fig. 2(d)) with non-negligible RTTs, by the time the VOQ length information in the request arrives at the output (selector), the actual VOQ length may have already been changed. If a grant is subsequently given to an empty VOQ, the slot will be wasted. To address this problem, we adopt the approach in [10]: Each output is equipped with N counters for keeping track of the length of the N VOQs destining to it. Besides, each request message contains two additional bits, a new-packet-arrival flag and a grant-rejected flag. The refined three-phase operation at slot t is detailed below:

Request: For input i, if a packet of VOQ(i, j) was served in slot $t-1$ and VOQ(i, j) is non-empty, send a request to output j; otherwise send requests for all non-empty VOQs. If a packet destined to output k arrives at current slot, the request sent to k should have the new-packet-arrival flag set. If a grant in the previous slot was not accepted, the request sent should have the grant-rejected flag set.

Grant: For output j with R being the set of requests received, if there are requests with new-packet-arrival flag set or with grant-rejected flag set, increase the corresponding VOQ counters by one. If R contains request from input which was served by it in slot $t-1$ and its VOQ counter is nonzero, grant the request; otherwise grant the request whose VOQ counter is the largest. In case of a tie, select a winner randomly. Then decrease the VOQ counter corresponding to the granted request by one.

Accept: For input i, accept the grant with the longest queue length and a tie is broken randomly. The remaining grants are rejected and recorded.

From the above operations, we can see that our D-LQF makes an explicit effort in avoiding granting empty VOQs. When RTT is negligible, the above three-phase operation will degenerate to become the basic scheme.

C. Refinement 2: solving starvation problem under inadmissible traffic

For admissible traffic patterns, i.e. no oversubscribed input and output ports, D-LQF can guarantee a close-to-100% throughput performance (as can be seen from simulation results). Therefore, throughput-fairness among different flows is not an issue, not to mention the possible starvation problem.

However, D-LQF may cause starvation at some flows under an inadmissible traffic pattern. For example, an oversubscribed output j may be overwhelmed by packets from VOQ(i, j) such that VOQ(i, j) will never be empty. Then all other flows destined to j will be starved. To prevent starvation, a simple way is to set a limit (M) on the number of consecutive packets to be served from each flow. Once the number exceeds M, we change the arbitration of the corresponding output from longest queue first to round robin. Interestingly, it is shown in [8] that if M is set to a large enough value, e.g. $M = N^3$, the throughput performance of the switch will not be negatively affected. In the next section, we focus on admissible traffic patterns for performance evaluations. In this case, M is set to infinity.

IV. Simulation Results

In this section, we study the delay-throughput performance of our proposed single-iteration scheduling algorithm D-LQF under two scenarios: negligible and non-negligible RTT (between linecards). When RTT is negligible, the following scheduling algorithms are implemented for comparison:

- Simple single-iteration scheduling algorithms: iSLIP-1 and iLQF-1 [5] (i.e. with single iteration).
- Tailor-made single-iteration scheduling algorithms: SRR [7] and π-RGA [8].
- Single-iteration algorithms with exhaustive service: EDRRM [12], E-iSLIP [13] and ELQF (the exhaustive service extension of LQF).

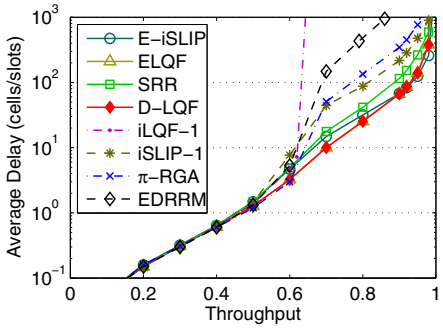

Fig. 4. Uniform Bernoulli traffic, RTT = 0.

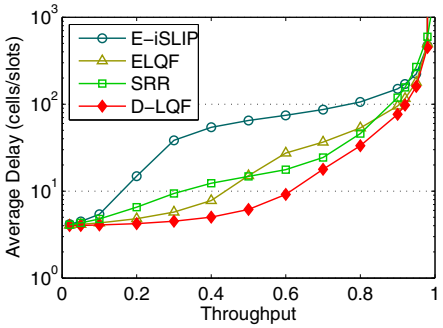

Fig. 5. Uniform Bernoulli traffic, RTT = 4.

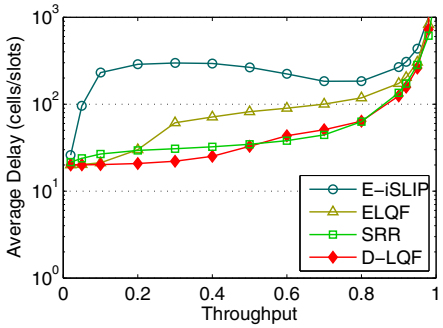

Fig. 6. Uniform Bernoulli traffic, RTT = 20.

When RTT is significant, we compare our D-LQF with the following fully distributed scheduling algorithms: SRR, E-iSLIP and ELQF. Note that we have extended the original E-iSLIP and ELQF for pipelined distributed implementation.

For brevity, the simulation results presented below are based on a 32×32 switch. Three admissible traffic patterns are considered: uniform Bernoulli, uniform bursty and hot-spot. The buffer size at each VOQ is set large enough for avoiding buffer overflow. Besides, the RTT values used in this section are all given in time slots.

A. Uniform Bernoulli Traffic

Uniform traffic is generated as follows. At every time slot for each input, a packet arrives with probability p (input load) and destines to each output with equal probability. From Figs. 4, 5 and 6, we can see that our D-LQF consistently outperforms all other algorithms by yielding the best delay performance.

When RTT = 0, we can see from Fig. 4 that π-RGA is worse than iSLIP and SRR. This indicates that SRR's notion of preferred inputs/outputs favors uniform traffic better than π-RGA's strong/weak request mechanism. It is also interesting to note that algorithms with exhaustive service are clear winners under all loading condition. This is because by leveraging the efforts over multiple time slots, they tend to yield a maximal matching in each time slot. Notably, EDRRM shows significant performance degradation when $p > 0.6$. (Note that throughput is equivalent to input load p when there is no packet loss due to buffer overflow.) This is because in EDRRM a non-engaged input only sends a single request in the request phase, so the input takes more time slots to establish a match with an output.

With non-negligible RTTs between linecards, Fig. 5 and 6 show that as compared with E-iSLIP and ELQF, D-LQF avoids incorrectly granting empty VOQs and thus significantly reduces wasted grants. Besides, we can see that ELQF outperforms E-iSLIP, which indicates that serving the longest VOQ first leads to a more "stable" maximal matching than serving VOQs in the round-robin order. We also notice that the gap between our D-LQF and SRR becomes narrower with RTT; this is because the scheduling overhead of D-LQF grows with RTT whereas for SRR is somewhat fixed.

B. Uniform Bursty Traffic

Bursty arrivals are modeled by an ON/OFF traffic model. In the ON state, a packet arrival is generated in every time slot. In the OFF state, no packet arrivals are generated. Packets of the same burst have the same output and the output for each burst is uniformly distributed. Given the average input load of p and average burst size s, the state transition probabilities from OFF to ON is $p/[s(1-p)]$ and from ON to OFF is $1/s$. Without loss of generality, we set burst size $s = 32$ packets. Again, Fig. 7, 8 and 9 compare the delay-throughput performance of various scheduling algorithms with variable RTTs.

When RTT = 0 and from Fig. 7, our D-LQF outperforms SRR and π-RGA by a large margin at high load. At $p = 0.8$, the average delay of using D-LQF is 176 time slots, whereas for SRR is more than 2300, and π-RGA is 241. This shows that π-RGA (and algorithms with exhaustive service) favor bursty traffic pattern more than SRR does. Similar performance trend is shown in Fig. 8 and 9 when RTT > 0, and the performance degradation of SRR is significant when $p > 0.6$.

C. Hot-Spot Traffic

Packets arriving at each input port in each time slot follow the same independent Bernoulli process with probability p. If a packet arrives at input i, it goes to output j ($j = i$) with probability $1/2$; goes to other outputs with equal probability $1/[2(N-1)]$.

From Fig. 10, 11 and 12, we can see that SRR saturates at around $p = 0.75$, whereas our D-LQF can obtain a close-to-100% throughput. Fig. 10 also shows that D-LQF and ELQF outperform E-iSLIP. This supports our analysis in Section III that serving longest VOQ first has smaller delay than serving VOQs in other manners.

From Fig. 11 and 12 we can see that as the value of RTT increases, E-iSLIP and ELQF suffer from incorrectly granting empty VOQs. Therefore, a wider performance gap with our D-LQF is observed at high input load and large RTTs. This shows that our D-LQF is indeed effective in identifying the most critical VOQs to serve first.

V. Conclusion

In this paper, we proposed a fully distributed single-iteration scheduling algorithm for input-queued switches called Dis-

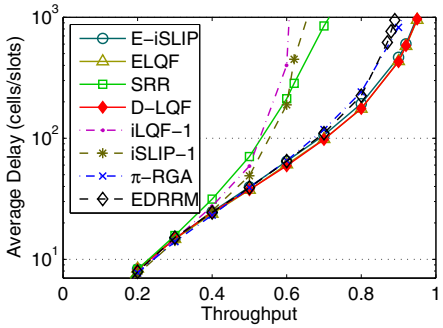

Fig. 7. Uniform bursty traffic, RTT = 0.

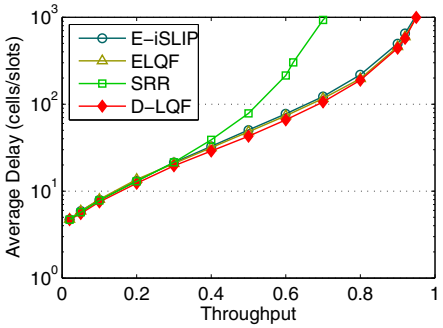

Fig. 8. Uniform bursty traffic, RTT = 4.

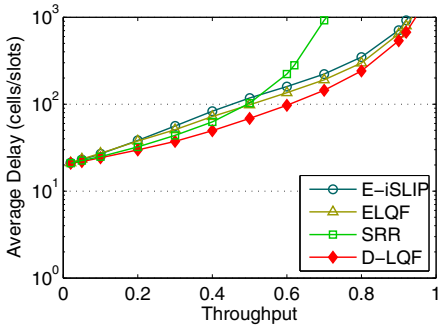

Fig. 9. Uniform bursty traffic, RTT = 20.

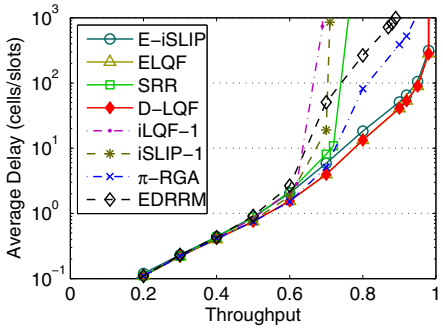

Fig. 10. Hotspot traffic, RTT = 0.

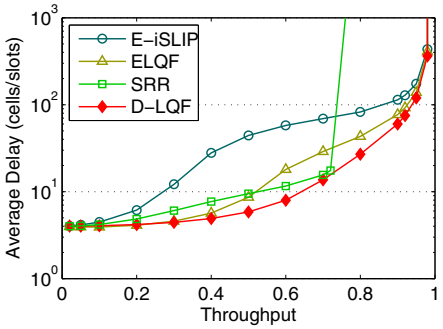

Fig. 11. Hotspot traffic, RTT = 4.

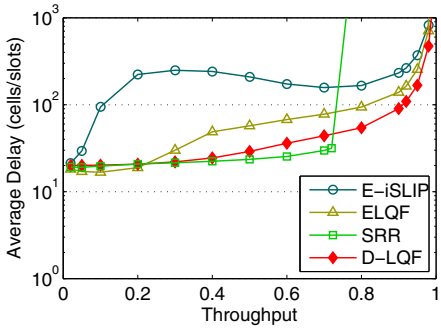

Fig. 12. Hotspot traffic, RTT = 20.

tributed Longest Queue First (D-LQF). D-LQF adopts exhaustive service policy for scheduling the longest queue first. To avoid granting an empty VOQ the chance to send a packet (due to the RTT between linecards), each output keeps track of the lengths of all VOQs destined to it. As compared with other scheduling algorithms, simulation results showed that our D-LQF gives the best delay-throughput performance.

REFERENCES

[1] H. J. Chao and B. Liu, *High Performance Switches and Routers*. John Wiley & Sons, Inc, 2007, ch. 7, pp. 225–230.

[2] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, 1993.

[3] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, 1999.

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

[5] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, University of California, Berkeley, 1995.

[6] C. Minkenberg, F. Abel, and E. Schiattarella, "Distributed crossbar schedulers," in *Proc. Workshop High Performance Switching and Routing*, 2006.

[7] A. Scicchitano, A. Bianco, P. Giaccone, E. Leonardi, and E. Schiattarella, "Distributed scheduling in input queued switches," in *Proc. IEEE Int. Conf. Communications ICC '07*, 2007, pp. 6330–6335.

[8] S. Mneimneh, "Matching from the first iteration: An iterative switching algorithm for an input queued switch," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 206–217, 2008.

[9] K. F. Chen, E. H. M. Sha, and S. Q. Zheng, "Fast and noniterative scheduling in input-queued switches: Supporting QoS," *Comput. Commun.*, vol. 32, no. 5, pp. 834–846, 2009.

[10] C. Minkenberg, "Performance of i-SLIP scheduling with large round-trip latency," in *Proc. HPSR High Performance Switching and Routing Workshop*, 2003, pp. 49–54.

[11] H. Takagi, "Queueing analysis of polling models: an update," in *Stochastic Analysis of Computer and Communication Systems*, H. Takagi, Ed. New York, NY, USA: Elsevier Science Inc., 1990, pp. 267–318.

[12] Y. Li, S. Panwar, and H. J. Chao, "The dual round robin matching switch with exhaustive service," in *Proc. Merging Optical and IP Technologies High Performance Switching and Routing Workshop*, 2002, pp. 58–63.

[13] Y. Li, S. S. Panwar, and H. J. Chao, "Exhaustive service matching algorithms for input queued switches," in *Proc. HPSR High Performance Switching and Routing 2004 Workshop*, 2004, pp. 253–258.