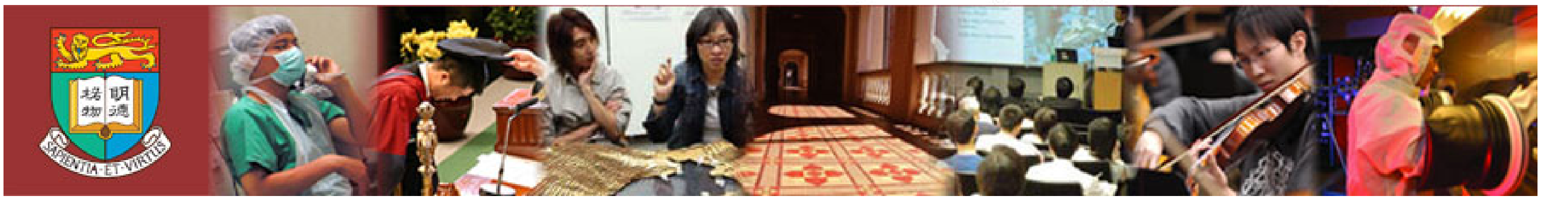

Minimizing the Communication Overhead of Iterative Scheduling Algorithms for Input-queued Switches

Bing Hu[1], Kwan L. Yeung[2], and Zhaoyang Zhang[1]

[1]Depart. of Information Science and Electronic Engineering
Zhejiang University
Hangzhou, PRC
Email: {binghu, ning_ming}@zju.edu.cn

[2]Depart. of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC
Email: kyeung@eee.hku.hk

***Abstract*—Communication overhead should be minimized when designing iterative scheduling algorithms for input-queued packet switches. In general, the overall communication overhead is a function of the number of iterations required per time slot (*M*) and the data bits exchanged in an input-output pair per iteration (*B*). In this paper, we aim at maximizing switch throughput while minimizing communication overhead. We first propose a single-iteration scheduling algorithm called Highest Rank First (HRF). In HRF, the highest priority is given to the preferred input-output pair calculated in each local port at a RR (Round Robin) order. Only when the preferred VOQ(*i,j*) is empty, input *i* sends a request with a rank number *r* to each output. The request from a longer VOQ carries a smaller *r*. Higher scheduling priority is given to the request with a smaller *r*. To further cut down its communication overhead to 1 bit per request, we design HRF with Request Compression (HRF/RC). The basic idea is that we transmit a single bit code in request phase. Then *r* can be decoded at output ports from the current and historical codes received. The overall communication overhead for HRF/RC becomes 2 bits only, i.e. 1 bit in request phase and 1 bit in grant phase. We show that HRF/RC renders a much lower hardware cost than multi-iteration algorithms and a single-iteration algorithm π-RGA [11]. Compared with other iterative algorithms with the same communication overhead (i.e. SRR [10] and 1-iteration iSLIP [6]), simulation results show that HRF/RC always produces the best delay-throughput performance.**

Keywords- Input-queued switch; single-iteration scheduling algorithm; iSLIP

I. INTRODUCTION

Input-queued switches suffer from the well known problem of Head-of-Line (HOL) blocking. This limits the maximum throughput of an input-queued switch to just 58.6% under uniform traffic [1]. To eliminate the HOL blocking, Virtual Output Queue (VOQ) is proposed [2], where each input port maintains a separate queue for each output (Fig. 1). A centralized scheduler is needed to maximize the throughput of a VOQ switch. The scheduling problem is equivalent to the matching problem in a bipartite graph [3]. It is found that the Maximum Weight Matching (MWM [4]) algorithm can guarantee 100% throughput. However, MWM algorithm has a high time complexity of $O(N^3 \cdot \log N)$. Maximal Size Matching (MSM) algorithms with lower computation overheads are then proposed. Various efficient implementations of MSM are also designed. Among them, the approach of using iterative scheduling algorithms is very popular due to the use of massive parallel processing [5-9]. However, the parallel processing is at the cost of higher communication overheads between input and output ports.

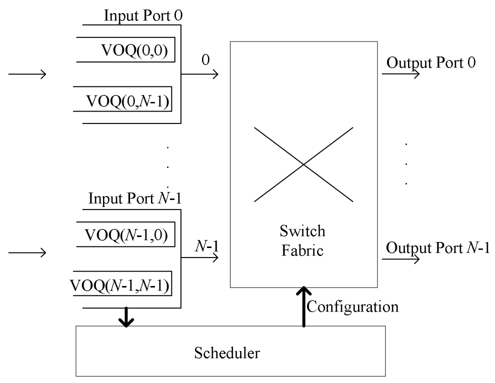

Fig. 1: An input-queued switch with a centralized scheduler

To this end, the recently proposed SRR (Synchronous Round Robin [10]), π-RGA (π-Request Grant Accept [11]) and SRA (Single Round-robin Arbitration [12]) utilize a single-iteration algorithm to construct switch configuration. In this paper, we propose a single-iteration scheduling algorithm for input-queued switches as well, called HRF (Highest Rank First). HRF always gives the highest priority to the preferred input-output pair calculated in each local port at a RR order. Only when the preferred VOQ(*i,j*) is empty, input *i* sends a request with a rank number *r* to each output. In HRF, *r* ($0 \le r \le 3$) is maintained at each VOQ to indicate its rank order among an input port according to its queue size. The request from a longer VOQ carries a smaller *r*. Higher scheduling priority is given to the request with a smaller *r*. To further cut down the communication overhead, we compress the request bits required by taking the past request into account. We call the resulting algorithm HRF/RC (HRF with Request Compression), whose overall communication overhead per input-output pair is 2 bits only. We show that the implementation complexity of HRF/RC is much lower than multi-iteration algorithms and a single-iteration algorithm π-RGA [11]. Compared with other iterative scheduling algorithms with the same communication overhead (SRR [10] and 1-iteration iSLIP [6]), HRF/RC can

This work was supported in part by Zhejiang Provincial Natural Science Foundation of China (No. Y1100388), Zhejiang Provincial Public Technology Research of China (No. 2010C31071), Fundamental Research Funds for the Central Universities (No. 2010QNA5032), National Science and Technology Major Project (No. 2011ZX03003-003-03).

provide the better delay-throughput performance under different traffic patterns.

The rest of this paper is organized as follows. In the next section, we review the existing efforts on single-iteration scheduling algorithms. Our single-iteration algorithm HRF is detailed in Section III and we further cut down the communication overhead of HRF in Section IV. In Section V, the delay-throughput performance of our design is compared with other approaches by simulations. Finally, we conclude the paper in Section VI.

II. Related Work

Before we review the existing related work, we introduce some definitions. An admissible traffic pattern [13] is defined as its traffic matrix satisfies

$$\sum_i \lambda_{ij} \leq 1, \quad \sum_j \lambda_{ij} \leq 1, \qquad (1)$$

where λ_{ij} is the mean packet arrival rate to VOQ(i,j). Weight of a VOQ refers usually, but not restricted to, the length (number of packets in backlog) of a VOQ [14]. Matching weight is the sum of the weight of all VOQs that have been matched to the outputs in that time slot. Matching size is the number of VOQs that have been matched to outputs in a switch configuration. We also define a precise measurement to evaluate the communication overhead in iterative algorithms. Specifically, we focus on the bits exchanged per input-output pair,

$$M \times B \qquad (2)$$

where M is the number of iterations required per time slot and B is the data bits exchanged in an input-output pair per iteration. Note that iterative scheduling algorithms conduct parallel processing. As far as one port concerned, the time consumed and hardware required for sending/receiving a single request are the same as that of N requests in patellar. As compared with the bits exchanged per port, the bits per input-output pair and (2) are more accurate for describing the overhead of parallel processing in iterative algorithms.

In iterative algorithms, one more iteration inevitably raises the scheduling time, which makes the multi-iteration algorithms not scalable for high speed routers/switches design. To this end, the recently proposed work [10-12] only utilize a single iteration to construct a switch configuration, i.e. $M = 1$ in (2). In general, there are three phases in a single-iteration algorithm: *request*, *grant* and *accept*. In request phase, input ports send matching requests to output ports. In grant phase, each output port grants at most one request received. Finally, in accept phase, every input port accepts at most one grant.

In SRR [10], each input port only issues a single request among its non-empty VOQs. For any input port i, the preferred request to output port j is calculated by $j = (i+t) \bmod N$, whereas t is current time slot. If the preferred VOQ is empty, then the longest one is selected. Each output j also has a preferential input i to grant based on the same cycle $j = (i+t) \bmod N$. If the preferred input request does not arrive, one request is randomly selected to grant. Therefore, the overall communication overhead (per input-output pair) for SRR is 2 bits only, i.e. 1 bit in request phase and 1 bit in grant phase respectively. The major problem with SRR is that it can only achieve high throughput performance under uniform traffic.

In π-RGA [11], there is a timer C(i,j) to record the last time when VOQ(i,j) transforms from empty to non-empty. Assumed that at time slot t, packet from VOQ(i,k) is served/transmitted, then at time slot t+1, input port i differentiates all its non-empty VOQs to *strong* or *weak* based on the following rules. If C(i,j) ≤ C(i,k), then VOQ(i,j) ∈ {strong} and else VOQ(i,j) ∈ {weak}. After such classification, if {strong} is empty but {weak} not, remove all VOQs in {weak} to {strong}. Like as other iterative algorithms, π-RGA can also be divided into three phases. In request phase, each input issues requests for all its non-empty VOQs. The VOQ status (strong or weak) and the value of C(i,j) are indicated in every request. In grant phase, priority is given to strong requests. If two or more strong requests arrive at an output port, the one with the minimal value of C(i,j) is granted. If there are only weak requests arriving, also choose the one with the minimal value of C(i,j). In accept phase, among the set of grants received by input port i, VOQ with the minimal value of C(i,j) is accepted. As compared with SRR, π-RGA can achieve higher throughput, especially under non-uniform traffic. Although π-RGA is a single-iteration scheme too, its communication overhead is severe

$$\log G + 2 \qquad (3)$$

where G is the maximum value for timer C(i,j).

In SRA [12], each output port j maintains a FIFO status queue for the non-empty VOQ(i,j)s (i =0,1,…N-1). The output j always chooses the HOL of the status queue, say it VOQ(i,j), to send a grant. Afterwards, output j removes VOQ(i,j) from HOL to the tail of the status queue. Upon receiving multiple grants, an input port accepts all grants and is matched to the corresponding outputs. If some VOQ(i,j) is to become empty after scheduling, input i sends a status signal to output j, and j will delete VOQ(i,j) from its status queue. The input i also sends status information to output j when any VOQ(i,j) changes from being empty to having a packet arrived. SRA finds a maximum input/output matching in a single iteration. But it allows one input port to send multiple packets to different outputs during a single time slot. In fact, this is one kind of speedups, which is hard to implement for high line rate.

III. Highest Rank First

In this section, we first dedicate our efforts on designing a single-iteration algorithm, named HRF (Highest Rank First). Then we explain why HRF is superior than other single-iteration algorithms.

A. HRF

HRF is designed as follows. Any VOQ is only possible in one of 4 statuses, i.e. *empty*, *non-empty*, *secondly longest* and *longest*, based on its queue size. These statuses are differentiated by a rank number r ($0 \leq r \leq 3$)

- If $r = 0$, VOQ(l,k) is the longest VOQ at input port l
- If $r = 1$, VOQ(l,k) is the secondly longest at input port l
- If $r = 2$, others but VOQ(l,k) is non-empty
- If $r = 3$, VOQ(l,k) is empty

If more than one VOQ is capable of the same longest queue length, then randomly evaluate one with $r = 0$, another with $r = 1$ and others with $r = 2$. At any case, there are at most one VOQ with $r = 0$ and one with $r = 1$ at each input port. Similar as other iterative algorithms, HRF is also composed of three phases.

Request: For any input port i, the preferred request to output port j is calculated by $j = (i+t) \bmod N$, whereas t is current time slot. If the preferred VOQ(i,j) is not empty, input i sends a single request to output j. Otherwise, input i issues requests for all its non-empty VOQs. The rank number r is indicated in every request sent.

Grant: Each output port j also has a preferential input port i to grant based on the same cycle $j = (i+t) \bmod N$. For any output j, the preferred input i is given the highest priority. If the preferred input request does not arrive, output j grants a request with the minimal r. When there are two or more requests with the same minimal r, select one uniformly at random.

Accept: Among the set of grants received by input port i, the preferred input-output pair is given the highest priority. If the preferred VOQ is empty, a grant to the VOQ with the minimal r is accepted. When there is more than one granted VOQ with the same minimal r, honor one with the longest queue size.

B. *Discussion*

In single-iteration algorithm π-RGA [11], output port j intends to grant the request coming from input port i, where VOQ(i,j) has the maximum weight among all requests received by output j. To simplify, we name this principle as OWF (Output Weight First). In OWF, the grant from output j would not be accepted as if input port i receives more weighted grants from other outputs. For example in π-RGA [11], assumed the request from *strong* VOQ(l,k) has the minimal value of C(l,k) among all requests to output port k. So output k gives the grant to input port l. At the same time slot, input l also receives another grant from output port h. Note that even though C(l,k) is the minimal one among C(i,k)s ($i = 0,1\ldots N-1$), it is still possible that C(l,k) > C(l,h). Then input l accepts output h and ignores the grant from output k. Consequently, the grant from output k is wasted and the size of matching is decreased. We can see that OWF/π-RGA seeks more matching weight but sacrifices the matching size of single-iteration algorithm.

How to reach more matching size and matching weight simultaneously in a single-iteration algorithm? Note that no matter how many requests received, each output port can only send a single grant per iteration. Then a single-iteration algorithm can just generate total N precious grants. The issue of seeking more matching weight and matching size can be converted to grant a more weighted request and avert wasting grants respectively. In other words, output port j should send grant to input port i which can accept j most likely and has more weight at the same time.

In HRF, output port j grants the request from input port i where VOQ(i,j) has the maximum weight among input i. To simplify, we call this principle as IWF (Input Weight First). Unlike OWF, any output j in IWF can make sure that its grant to input i would be accepted as long as VOQ(i,j) has the maximum weight among input i. For example in HRF, if an output port grants a request with $r = 0$, such grant is predestined to be accepted for sure. Therefore, IWF avoids wasting grants as best as it can and gives grant to the most weighted request (among an input). As compared with OWF/π-RGA, IWF/HRF yields the similar matching weight but more matching size in a single-iteration algorithm.

Like as SRR [10], HRF always gives the highest priority to the preferred input-output pair calculated by $j = (i+t) \bmod N$. Only when the preferential VOQ(i,j) is empty, input port i sends requests to other output ports. In essence, “two” iterations are preceded, i.e. the “first” local iteration for the preferred input-output pair and the “second” true iteration for the unmatched ports after the preferential matching. In summary, the “2-iteration” nature and IWF principle make HRF superior than other single-iteration algorithms.

IV. HIGHEST RANK FIRST WITH REQUEST COMPRESSION

Multi-iteration algorithm iSLIP [6] has been widely adopted in commercial products (e.g. Cisco 12000 series routers [15]). When implementing iSLIP, a control circuit connects each VOQ to the centralized scheduler for sending requests [6]. This successful experience has been applied to implement other iterative algorithms. The only requirement is 1-bit sending/receiving during request phase as the control circuit favors a single bit transmitted. Recall that HRF sends $\log r = 2$ bits during its request phase. To reutilize the simple and mature implementation technique from iSLIP, we compress HRF to send 1-bit request only.

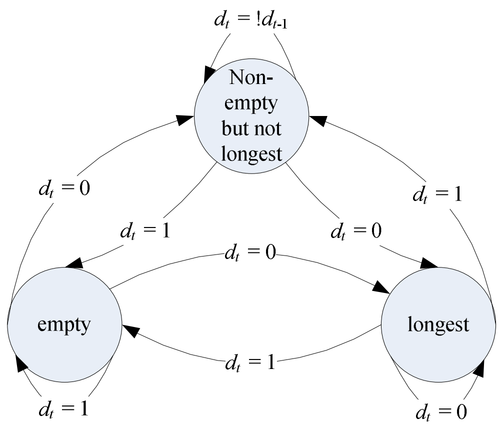

Fig. 2: Encode for statuses transition in VOQ

A. *HRF with Request Compression*

The basic idea is that we transmit a single bit code in request phase. Then r can be decoded at output ports from the current and historical codes received. To distinguish with original HRF, we call the resulted algorithm as HRF/RC (Highest Rank First with Request Compression). Unlike HRF, HRF/RC only maintains 3 statuses for all VOQs, i.e. *empty*, *non-empty* and *longest*. Like HRF, HRF/RC is also composed of 3 phases.

Request: Let d_t be the single bit sent at time slot t during request phase. For any input port i, if the preferred VOQ(i,j) is not empty, where j is calculated by $j = (i+t) \bmod N$, input i sends “0” to output j and “1” to other outputs. Otherwise, input *i* sends 1-bit d_t to each output port based on the statuses transition in Fig. 2. For example, if VOQ(*i*,*j*) becomes non-empty from empty, input *i* sends $d_t = 0$ to output *j*.

Grant: Output port *j* recorded d_{t-1} (received at previous time slot *t*-1) to decode the bit d_t received at current time slot *t*. If $d_{t-1}\, d_t = 00$, $r = 0$; $d_{t-1}\, d_t = 10$, $r = 1$; $d_{t-1}\, d_t = 01$, $r = 2$; $d_{t-1}\, d_t = 11$, $r = 3$. Each output port *j* always gives the highest priority to its preferred input port *i* based on the same cycle $j = (i+t) \bmod N$, as long as output *j* receives $d_t = 0$ from input *i*. Otherwise, output *j* grants 1 bit to the request with the minimal *r* received.

Accept: Among the set of grants received by input port *i*, the preferred input-output pair is given the highest priority. If the preferred VOQ is empty, a grant to the VOQ with the minimal *r* is accepted. Note that no bit is exchanged at this phase.

B. Discussion

In total, HRF/RC is a single-iterative algorithm with 2-bit communication overhead, i.e. 1 bit in request phase and 1 bit in grant phase. Then its request phase can be finished within a single trip using the control circuit adopted by iSLIP [6]. Compare with HRF, HRF/RC cuts down the 1/3 communication overhead but provides the similar delay-throughout performance. Nevertheless, simulation results (in Section V) show that both HRF and HRF/RC are superior than other iterative algorithms with the same communication overhead.

Besides communication overhead, HRF/RC also reduces the computation complexity of HRF. Any VOQ of HRF is in one status of empty, non-empty, secondly longest and longest. The complicated sorting operation for all *N* VOQs may be required to find the secondly longest one. But HRF/RC only maintains three statuses of empty, non-empty, and longest for VOQs. As such, we just need to carry out an simple algorithm to find the longest VOQ among an input port. For example, we can use Quasi-LQF [16], a very efficient suboptimal LQF algorithm requiring only a single comparison per time slot. We can see that HRF/RC renders a much lower hardware cost than multi-iteration algorithms, π-RGA and HRF.

V. Performance Evaluations

In this section, we study the performance of iterative algorithms under the comparable overhead. To result the similar 2-bit communication as HRF/RC, the multi-iteration algorithms iSLIP [6] and iLQF [7] are run at one iteration. The single-iteration algorithms SRR [10] and π-RGA [11] are set with $M = 1$ and $G = 16$ in (2) and (3). Therefore, the communication overheads per input-output pair are 2 bits for SRR and iSLIP, 6 bits for π-RGA and iLQF. Output-queued switch is also implemented as a lower bound. Although we only present simulation results for switch with size $N = 32$ below, the same conclusions and observations apply for other switch sizes.

A. Uniform Traffic

Uniform traffic is generated as follows. At every time slot for each input, a packet arrives with probability *p* (input load *p*) and destines to each output with equal probability. From Fig. 3, we can see that HRF and HRF/RC can obtain up to 100% throughput and the best delay performance among all iterative algorithms. The curves of HRF and HRF/RC almost overlap with each other. This verifies that the encode/decode in HRF/RC is quite efficient to track the value of *r* in HRF. Compared with π-RGA, HRF/RC gives significantly smaller delay. When $p = 0.8$, π-RGA requires 139.7 time slots, and HRF/RC only 13.6, cutting down the delay by more than 12 times. When *p* is reasonably large ($p > 0.6$), our HRF/RC also beats SRR.

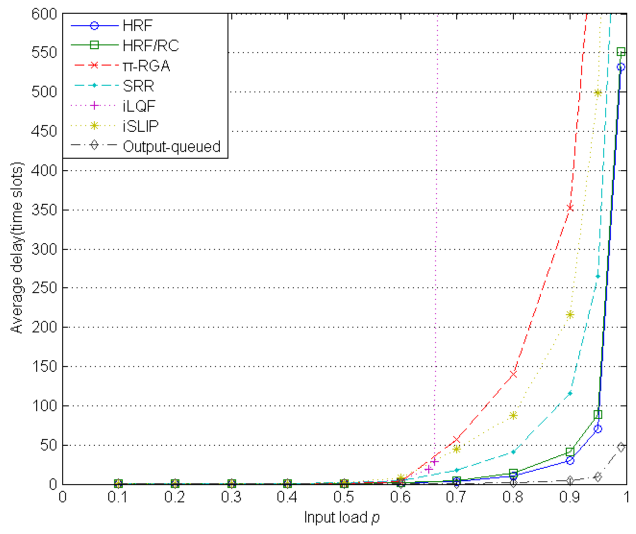

Fig. 3: Delay vs input load, under uniform traffic

B. Uniform Bursty Traffic

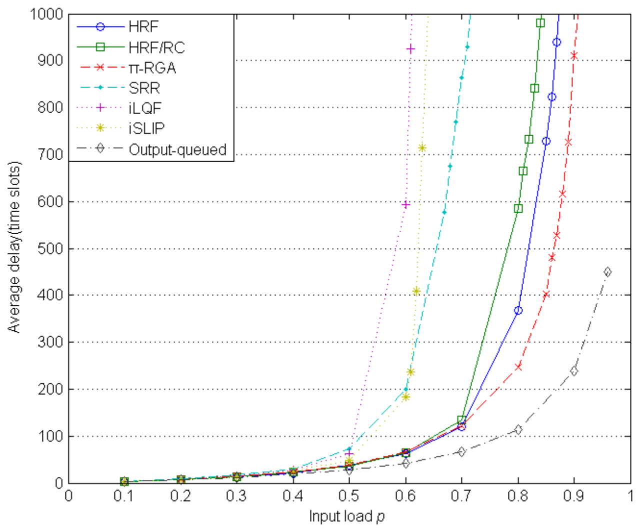

Fig. 4: Delay vs input load, under uniform bursty traffic

Bursty arrivals are modeled by the ON/OFF traffic model, which is a special instance of the two-state Markov-modulated Bernoulli process [17]. In the ON state, a packet arrival is generated in every time slot. In the OFF state, there are no packet arrivals. 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 $s = 30$ packets.

From Fig. 4, we can see that delay builds up quickly with input load. iSLIP and iLQF only achieve 60% throughput but HRF and HRF/RC obtain 80% throughput. When $p > 0.7$, there is a tiny performance gap between HRF and HRF/RC. This is because HRF/RC cannot accurately decode the ranks of VOQs under such highly bursty traffic. Nevertheless, HRF/RC consistently shows superior to SRR even when traffic is light. For example at $p = 0.6$, with SRR packets experience an average delay of 200 time slots, whereas for HRF/RC is just 63.7. We can also see that HRF/RC outperforms π-RGA when $p < 0.7$. When $p > 0.7$, π-RGA can catch the bursty flows so it shows the less delay than HRF/RC. But it should be noted that π-RGA is not scalable/practical because of its heavier communication overhead.

C. Hot-spot Traffic

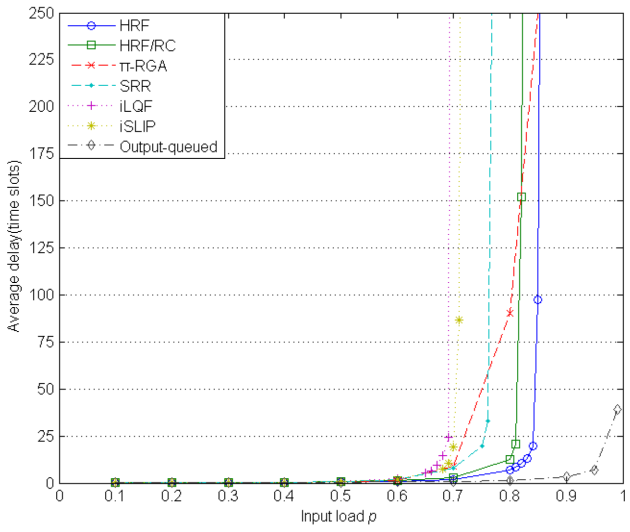

Fig. 5: Delay vs input load, under hot-spot traffic

We assume packets arriving at each input port in each time slot follow the same independent Bernoulli process with probability p. Hot-spots are generated as follows. For input port i, packet goes to output $i+N/2 \bmod N$ with probability 0.5, and goes to other outputs with the same probability $1/[2(N-2)]$. From Fig. 5, again we can see that iSLIP and iLQF only achieve 70% throughput but HRF and HRF/RC provide 80% throughput. HRF consistently shows superior to SRR and π-RGA. Even HRF/RC outperforms π-RGA when $p < 0.82$.

In summary, HRF and HRF/RC yield the best delay-throughput performance under uniform and hot-spot traffic. Only π-RGA beats HRF/RC under bursty traffic, but this is at the cost of 3 times of communication overhead. Although HRF/RC renders a much lower hardware cost than HRF, it provides the similar delay-throughout performance as HRF.

VI. CONCLUSIONS

In this paper, we proposed a single-iteration scheduling algorithm for high speed switches, named HRF. HRF always gives the highest priority to the preferred input-output pair calculated at each local port. Only when the preferred VOQ(i,j) is empty, input i sends a request with a rank number r to each output. In HRF, r ($0 \le r \le 3$) is maintained at each VOQ to indicate its rank order among the input port according to the queue size. The smaller r stands for the longer VOQ, and thus is given the higher priority in scheduling. To further cut down communication overhead, we compressed the request bits of HRF, called HRF/RC. The resulted design only makes use of 1 bit in request phase and 1 bit in grant phase. Then HRF/RC can be implemented at a much lower cost than HRF. Simulation results showed that both HRF and HRF/RC consistently yield the better delay-throughput performance than other iterative algorithms with the same hardware complexity (i.e. SRR [10] and 1-iteration iSLIP [6]).

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*, Vol. 35, pp. 1347 – 1356, Dec. 1987.

[2] Y. Tamir and G. L. Frazier, “High-performance multi-queue buffers for VLSI communications switches,” *Proceeding 15th Annual Symposium Computer Architecture*, pp. 343 – 345, June 1988.

[3] G. Chartrand, “Introductory graph theory,” *New York Dover*, pp. 116, 1985.

[4] N. McKeown, V. Anantharam and J. Walrand, “Achieving 100% throughput in an input queued switch,” *IEEE INFOCOM*, April 1996, San Francisco, USA.

[5] T. Anderson, S. Owicki, J. Saxes and C. Thacker, “High speed switch scheduling for local area networks,” *ACM Transactions on Computer Systems*, Vol. 11, pp. 319 – 352, 1993.

[6] N. McKeown, “The iSLIP scheduling algorithm for input-queued switches,” *IEEE/ACM Transactions on Networking*, Vol. 7, No. 2, pp. 188 – 201, April 1999.

[7] N. McKeown, “Scheduling algorithms for input-queued cell switches,” *PhD. Thesis*, University of California at Berkeley, 1995.

[8] Y. Li, S. Panwar and H. J. Chao, “Saturn: a terabit packet switch using dual round robin,” *IEEE Communication Magazine*, Vol. 38, No. 12, pp. 78–84, Dec. 2000.

[9] E. Leonardi, M. Mellia, F. Neri and M. A. Marsan, “On the stability of input-queued switches with speed-up,” *IEEE/ACM Transitions Networking*, Vol. 9, No. 1, pp. 104–118, Feb. 2001.

[10] A. Scicchitano, A. Bianco, P. Giaccone, E. Leonardi and E. Schiattarella, “Distributed scheduling in input queued switches” *IEEE ICC 2007*, June 2007, Glasgow, Scotland.

[11] S. Mneimneh, “Match form the first iteration: an iterative switching algorithm for input queued switch,” *IEEE/ACM Transitions on Networking*, Vol. 16, No. 1, pp. 206 – 217, Feb. 2008.

[12] K. Chen, S. Q. Zheng and E. H. M. Sha, “Fast and noniterative scheduling in input-queued switches: supporting QoS,” *Computer Communications*, Vol. 32, No. 5, pp. 834-846, March 2009.

[13] B. Hu and K. L. Yeung, “Feedback-based scheduling for load-balanced two-stage switches,” *IEEE/ACM Transactions on Networking,* Vol. 18, Issue. 4, pp. 1077-1090, Aug. 2010.

[14] D. Shah and M. Kopikare, “Delay bounds for approximate maximum weight matching algorithms for input queued switches,” *IEEE INFOCOM*, June 2002, New York, USA.

[15] http://www.cisco.com/en/US/products/hw/routers/ps167/index.html

[16] Y. S. Lin and C. B. Shung, “Quasi-pushout cell discarding,” *IEEE Communication Letter*, Vol. 1, No. 5, pp. 146–148, Sep. 1997.

[17] B. Hu and K. L. Yeung, “Load-balanced optical switch for high-speed router design,” *IEEE/OSA Journal of Lightwave Technology,* Vol. 28 , Issue. 13, pp. 1969-1977, July 2010.