On Iterative Scheduling for Input-Queued Switches With a Speedup of $2-1/N$

Bing Hu, *Member, IEEE*, Kwan L. Yeung, *Senior Member, IEEE*, Qian Zhou, and Chunzhi He

***Abstract*— An efficient iterative scheduling algorithm for input-queued switches, called round robin with longest queue first (RR/LQF), is proposed in this paper. RR/LQF consists of three phases: report, grant, and accept. In each phase, only a single-bit message per port is sent for reporting a packet arrival, granting an input for packet sending, or accepting a grant. In both the grant and accept phases, scheduling priority is given to the preferred input–output pairs first and the longest virtual output queuing (VOQ) next. The notion of the preferred input–output pair is to keep a global RR schedule among all the inputs and the outputs. By serving the preferred input–output pairs first, the match size tends to be maximized. By serving the longest VOQ next, the match weight is also boosted. When RR/LQF is executed for a single iteration (i.e., RR/LQF-1), we show by simulations that RR/LQF-1 outperforms all the existing single-bit-single-iteration scheduling algorithms. When RR/LQF is executed up to N iterations (i.e., RR/LQF-N), we prove that under any admissible traffic pattern, RR/LQF-N is stable with a speedup of $2-1/N$, where N is the switch size. To the best of our knowledge, this is the first work showing that an iterative scheduling algorithm is stable with a speedup less than 2. We then generalize RR/LQF to become a class of algorithms that have the same speedup bound of $2-1/N$. Efforts are then made to further reduce the implementation complexity of RR/LQF. To this end, the pipelined RR/LQF and RR/RR, a simpler variant of RR/LQF, are proposed.**

***Index Terms*— Input-queued switch, iterative scheduling algorithm, 100% throughput.**

I. Introduction

A. High-Speed Switch-on-a-Chip (SoC) Design

The ever-increasing demand for Internet bandwidth is further fueled by the ongoing shift to cloud computing. In a cloud computing architecture, services and data reside in shared data centers [1]–[3], and are accessed by users over the Internet. Huge amount of IP traffic will be transported between users and data centers, and between servers within a data center. In addition to IP traffic, most data centers have storage area networks for carrying data between hundreds of thousands of servers and disk arrays, and some data centers have high-performance computing interconnects for low-latency inter-process communications. To meet the need for high-speed and low-latency switching, it is highly desirable to have a unified switch fabric for all three types of traffic, IP, storage and computing.

In designing a high-performance switch, the notion of switch-on-a-chip (SoC) [4] is widely adopted, whereby the entire data plane of a switch is driven by a single chipset. In order to minimize I/O pins of a chipset, inter-chip communication is serial via a device called SerDes (serializer/deserializer). A SerDes consists of a transmitter and a receiver. The transmitter converts parallel data to serial for transmission and the receiver converts serial data back to parallel for processing. The speed of a port depends on the speed of a SerDes. The current state-of-art allows a SerDes to operate at 25 Gbps. Efforts are made to raise it to 100 Gbps. To obtain the speed of, e.g. 100 Gbps, a multi-lane approach [5] of running 4×25 Gbps SerDes in parallel is adopted. Assume that packets are of the same size and each time slot can accommodate one packet. On a 100 Gbps line carrying 64-byte packets, one time slot is about 5ns. This puts a huge pressure on SoC design.

The Broadcom BCM56850 chipset [6], also known as the Trident II, is adopted by some fastest switches on the market (e.g. Cisco Nexus 9000 [7] and Juniper QFX5100 [4]). BCM56960 or Tomahawk [8] is the successor of BCM56850. It allows up to 128 ports with each port/SerDes running at 25 Gbps. While the exact switch architectures of the Trident II and Tomahawk are not known, Cisco's custom ASIC, which is used in Cisco Nexus 5000 [9], is a 58×58 input-queued crossbar switch with an on-chip scheduler. The scheduler is likely a variant of iSLIP[1] [10], a classic iterative scheduling algorithm for input-queued switches.

B. Input-Queued Switch and Iterative Scheduling Algorithm

As compared to output-queued switch, input-queued switch based on crossbar switch fabric [11], [12] is more suitable for high-speed SoC implementation because of the reduced

Manuscript received January 20, 2015; revised July 13, 2015 and January 1, 2016; accepted February 1, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Mellia. This work was supported in part by the Zhejiang Provincial Public Technology Research Project under Grant 2015C31108, the Open Foundation of State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, under Grant SKLNST-2013-1-20, the National Basic Research Program of China (973 Green) under Grant 2012CB316000, and the National Natural Science Foundation of China under Grant 61401394.

B. Hu and Q. Zhou are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China, and also with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100088, China (e-mail: binghu@zju.edu.cn; qianzhou@zju.edu.cn).

K. L. Yeung and C. He are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: kyeung@eee.hku.hk; czhe@eee.hku.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2541161

[1]Due to its simplicity and good performance, iSLIP is by far the most widely cited and compared iterative scheduling algorithm. To the best of our knowledge, it is also the only iterative scheduling algorithm that has been commercially implemented.

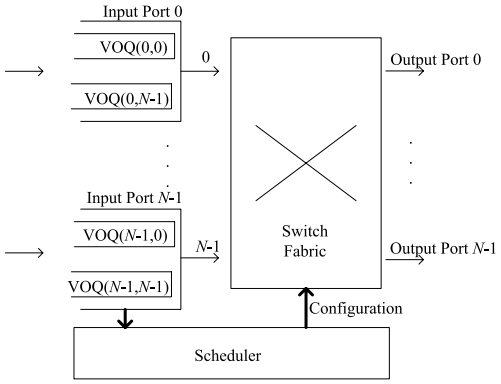


Fig. 1. An input-queued switch with a centralized scheduler.

memory bandwidth requirements – in each time slot at most one packet is sent/received by an input/output port. To eliminate the head-of-line (HoL) blocking [11] problem, input-queued switch adopts Virtual Output Queuing (VOQ) [12] as shown in Fig. 1. To maximize the switch throughput, a central scheduler is required for contention resolution when two or more packets destined to the same output in the same time slot. The scheduling problem is equivalent to the matching problem in a bipartite graph. A scheduling algorithm is stable if it guarantees 100% throughput (or non-blocking performance) for any admissible traffic pattern.[2] Scheduling algorithms such as maximum size/weight matching [13], [14] are stable, but they are too complicated for efficient hardware implementation.

As a result, sub-optimal algorithms for finding *Maximal* Size Match (MSM) are usually adopted. A matching is of maximal size if no input or output is left unnecessarily idle [10], [15], [16]. Finding an MSM is more efficient because it does not involve backtracking. Among various implementations of MSM, the approach of using iterative scheduling algorithms is widely accepted due to the feasibility of use of massive parallel processing [17], [18]. In general, each iteration of an iterative scheduling algorithm consists of three phases, request, grant and accept. In the request phase, input ports send matching requests to outputs. In the grant phase, each output selects one request to grant. In the accept phase, each input picks one grant to accept, and notifies the corresponding output not to participate in the subsequent iterations.

An iterative scheduling algorithm needs to execute up to N iterations in order to guarantee maximal size match. To the best of our knowledge, all existing iterative algorithms that have been proved to be stable [19], [20] require a speedup of 2, i.e. in each time slot an input/output is allowed to send/receive up to two packets. A lower speedup is always desirable because it can reduce the implementation cost. Another factor that affects the implementation cost of an iterative algorithm is the number of iterations to be executed. It is found that the increase in match size is diminishing with the number of iterations. To reduce the scheduling overhead (and thus lower the implementation cost), a single-iteration[3] scheduling algorithm can be adopted. Notably, if the round trip time between inputs and outputs is non-negligible [21], [22], single-iteration scheduling algorithm is a must. On the other hand, if only single bit messages are sent in each scheduling phase, the scheduling algorithm is a single-bit algorithm. Accordingly, a single-bit-single-iteration algorithm minimizes the scheduling overhead. (Please refer to Section II.A for a taxonomy of iterative scheduling algorithms.)

Besides input-queued switches, another popular approach in high-speed switch design is using load-balanced two-stage switch [23]–[25]. The main advantage of load-balanced two-stage switch is no central scheduler. The main disadvantage is the packet mis-sequencing problem [24]. Besides, the load-balanced switch suffers from the high delay performance under low to medium load (i.e. there exists a high "delay floor" due to the periodic switch configuration sequence). Load-balanced two-stage switch is an excellent approach of getting rid of central scheduler, but the main challenge today is still on how to make the central scheduler more efficient.

C. Our Contributions

It is our belief that the performance of an iterative scheduling algorithm hinges on to what extent we can benefit from two simple and generic schedulers, Round Robin (RR) and Longest Queue First (LQF). Notably, RR focuses on maximizing the match size and LQF targets at addressing the most critical VOQs. A good scheduler should try to get the best out of both, and more importantly, in a way that can adapt to the real-time traffic fluctuation. In this paper, a new single-bit iterative scheduling algorithm called Round Robin with Longest Queue First (RR/LQF) is proposed with the above in mind. If a single iteration is executed in each time slot, we refer to it as RR/LQF-1. It is a single-bit-single-iteration algorithm. If up to N iterations are executed for finding maximal size match in each time slot, we refer to it as RR/LQF-N. When the difference between the two is not important, we simply use RR/LQF. In this paper, our major contributions are:

a) RR/LQF-1 is the best performing algorithm among the class of single-bit-single-iteration scheduling algorithms.
b) RR/LQF-N is the first work showing that an iterative scheduling algorithm is stable with a speedup less than 2. It is then generalized to become a class of algorithms with the same speedup bound.
c) RR/LQF guarantees the max-min fair bandwidth allocation under certain traffic scenarios.
d) The complexity of RR/LQF is further reduced by the pipelined RR/LQF and RR/RR, a simpler variant of RR/LQF.

The rest of this paper is organized as follows. In the next section, we review the related work on iterative scheduling algorithm design. In Section III, RR/LQF is detailed. Its stability is studied in Section IV. In Section V, the delay-throughput performance of RR/LQF is compared with other

[2]A traffic pattern is admissible if there are no oversubscribed inputs and outputs. Please also refer to the definition in (3).

[3]A single iteration algorithm does not need the accept phase.

TABLE I
A TAXONOMY FOR ITERATIVE SCHEDULING ALGORITHMS

		Single-iteration	Multiple-iteration
Single-bit-request (SBR)	Single SBR	RR/LQF-1, SRR [21], DRR-1 [17], pDRR-1 [18], LQD-1 [19]	RR/LQF-*N*, DRR-*N*, pDRR-*N*, LQD-*N*
	Multiple SBRs	PIM-1 [15], iSLIP-1 [10], SRRR-1 [27]	PIM-*N*, iSLIP-*N*, SRRR-*N*
Multiple-bit-request (MBR)	Multiple MBRs	iLQF-1, iOCF-1 [16], π-RGA [22]	iLQF-*N*, iOCF-*N*

iterative scheduling algorithms by simulations. In Section VI, we examine the fairness performance of RR/LQF under different traffic scenarios. Some implementation issues are discussed in Section VII. We conclude the paper in Section VIII.

II. RELATED WORK

A. A Taxonomy for Iterative Scheduling Algorithms

A taxonomy for iterative scheduling algorithms is given in Table 1. If an iterative algorithm only executes one iteration in each time slot, it is a single-iteration algorithm; otherwise, it is a multiple-iteration algorithm. A single-iteration algorithm can be tailor-made, e.g. SRR [21], or simply executing a multiple-iteration algorithm (e.g. PIM [15] and iSLIP [10]) for a single iteration, i.e. PIM-1 and iSLIP-1. Understandably, a tailor-made single-iteration algorithm tends to give better performance.

In the request phase, an input can either send a single-bit-request (SBR) or a multiple-bit-request (MBR) to the corresponding output(s). An SBR usually indicates whether a VOQ is empty ("0") or backlogged ("1"). An MBR provides additional information about the VOQ status, e.g. the age of head-of-line packet in the VOQ (as in iOCF [16]), or the size of VOQ (as in iLQF [16]), or the last time when the VOQ transforms from empty to backlogged (as in π-RGA [22]). Note that we focus on request message size because messages for grant and accept are all single-bit.

In the request phase, an input can either send a single request to one of its backlogged VOQ outputs, or multiple requests, one for each backlogged VOQ output (or a subset of backlogged VOQ outputs [26]). Sending one request has the advantage of less communication overhead and less "distraction" at outputs (because after all, each output can only grant one request), but match size is in danger if requests from different inputs are sent to the same output. If all single requests are sent to distinct outputs (or as distinct as possible), such an adverse impact to the match size can be minimized.

B. Single-Bit-Single-Iteration Algorithms

In Table 1, single-bit-(request)-single-iteration algorithms consist of two sub-types, with single SBR per input and multiple SBRs per input.

PIM-1 [15], iSLIP-1 [10] and SRRR-1 [27] belong to the category of multiple SBRs. They differ in the input/output arbitrations adopted. PIM-1 picks up the winner randomly. iSLIP-1 selects the winner according to a Round Robin (RR) pointer. The recently proposed SRRR-1 is a *four*-phase scheduling algorithm. The extra/first phase is for an output to inform an input if it is pointed by an RR pointer (i.e. a preferred input). The idea is to desynchronize output pointers for maximizing match size. Simulations show that SRRR-1 performs better than other multiple SBRs algorithms, but at a rather steep cost of an extra phase.

DRR-1 [17], pDRR-1 [18], LQD-1 [19] and SRR [21] belong to the category of single SBR per input. The only difference between iSLIP-1 and DRR-1 is that an input of iSLIP-1 issues a single-bit-request for *each* backlogged VOQ output and that of DRR-1 transmits a single-bit-request to *one* of the backlogged VOQ outputs. In pDRR-1, DRR-1 is generalized to support multiple priorities. In LQD-1, each input sends one request based on a probability proportional to the VOQ length. If an output port receives several requests, LQD-1 selects the winner randomly.

SRR (Synchronous Round Robin [21]) is more closely related to our proposed RR/LQF. It is the first algorithm that adopts the notion of preferred input-output pairs, i.e. a synchronous round robin relationship. Consider an $N \times N$ input-queued switch in Fig. 1. In each time slot, each input is assigned a distinct preferred output. Without loss of generality, for input i at time slot t, its preferred output j is given by

$$j = (i + t) \bmod N \tag{1}$$

It can be easily seen that each input "prefers" each output exactly once in every N slots. When input i prefers output j, output j also prefers input i. This forms a preferred input-output pair.

In SRR, scheduling priority is given to the preferred input-output pairs first. In the request phase, if input i's preferred output j is backlogged, i.e. VOQ(i,j) > 0, input i sends a single-bit-request to output j. Otherwise, input i transmits a single-bit-request to its longest VOQ output. In the grant phase, if output j receives a request from its preferred input i, output j grants input i. Otherwise, output j randomly selects an input (request) to grant. It is worth noting that the preferred input-output relationship is pre-determined and it desynchronizes the choices made by respective inputs and outputs. The resulting improvement in match size and thus throughput is particularly significant under heavy uniform traffic load. Nevertheless, SRR's performance under non-uniform traffic is generally poor.

C. Single-Bit-Multi-Iteration Algorithms

In Table 1, single-bit-multi-iteration algorithms include PIM-N [15], iSLIP-N [10], SRRR-N [27], DRR-N [17], pDRR-N [18], and LQD-N [19], where the suffix "-N" indicates that the algorithm is to be executed up to N iterations until it finds the maximal size match. The single-bit-multi-iteration algorithms also consist of two sub-types, with single SBR per input or multiple SBRs per input. PIM-N, iSLIP-N

and SRRR-N belong to the category of multiple SBRs, while DRR-N, pDRR-N and LQD-N adopt single SBR.

D. Multi-Bit-Single-Iteration Algorithms

In a multi-bit-request (MBR) scheme, a VOQ can send requests that carry either the actual VOQ size (as in iLQF [16]) or the "age" of HoL packet (as in iOCF [16]). When they are restricted for executing a single iteration, we call them multi-bit-single-iteration algorithms. In Table 1, all existing multi-bit-single-iteration algorithms belong to the sub-type of multiple MBRs per input. Among them, if an output/input port receives more than one request/grant, iLQF-1 prefers the longest VOQ and iOCF-1 likes the VOQ with the oldest packet.

Unlike iLQF-1 and iOCF-1, π-RGA [22] is a tailor-made multi-bit-single-iteration algorithm. It uses a timer $T(i,j)$ to record the last time when VOQ(i,j) transforms from empty to backlogged. Assume that at time slot t, a packet from VOQ(i,k) is sent. At slot $t+1$, input port i identifies all non-empty VOQs as *strong* or *weak* based on the following rule: if $T(i,j) \leq T(i,k)$, VOQ$(i,j) \in$ {strong}; otherwise VOQ$(i,j) \in$ {weak}. If {strong} set is empty, remove all VOQs in {weak} to {strong}. In the request phase, each input issues requests for all its backlogged VOQs. The request contains the VOQ status (*strong* or *weak*) and the value of $T(i,j)$. In the grant phase, the strong request with the smallest $T(i,j)$ has the highest priority. If there are no strong requests, the weak request with the smallest $T(i,j)$ is granted. Finally in the accept phase, the grant with the smallest $T(i,j)$ is accepted. In general, π-RGA provides high throughput under non-uniform traffic, at the cost of higher communication overhead for carrying VOQ status and $T(i,j)$.

E. Multi-Bit-Multi-Iteration Algorithms

When iLQF and iOCF [16] are executed up to N iterations until they find the maximal size match, they become multi-bit-multi-iteration algorithms (see Table 1), denoted by iLQF-N and iOCF-N respectively. It is interesting to note that there is no single MBR scheme. This can be explained as follows. MBR generally carries the weight information (i.e. queue size or packet age). To justify the use of weight, grants should be selected judiciously. Using multiple MBRs allows grants to be picked from (up to) N^2 requests, whereas that of single MBR would be (up to) N requests only. The more candidates can make better use of the weight information. As a result, no single MBR scheme has been designed.

F. Stability Properties of Iterative Algorithms

It has been shown [19], [20] that a maximal size match algorithm is not stable, i.e. cannot guarantee 100% throughput for any admissible traffic pattern. Speedup is thus required for providing 100% throughput. A speedup of S can be achieved by operating the switch fabric at S times faster than each port rate such that the duration of a packet inside the fabric is S times shorter. For a 100 Gbps port, a 64-byte packet occupies a time slot of merely 5ns. A speedup of S will reduce the slot duration to $5/S$ ns. This imposes a huge challenge to switch chipset design. Obviously, a small S is highly desirable in lowering the implementation cost.

The stability properties of iterative algorithms are studied in [19] and [20] using the analytical techniques primarily based on Lyapunov functions. Their main result is that, for a wide class of iterative scheduling algorithms, stability is guaranteed by a speedup of 2. In this paper, we show that RR/LQF only needs a speedup of 2-1/N to ensure 100% throughput. RR/LQF is therefore the first iterative algorithm that can operate with a speedup less than 2.

III. Round Robin With Longest Queue First

A. Global RR and LQF Schedulers

A good iterative algorithm should maximize not only the match size but also the match weight. A round robin (RR) scheduler over all input and output ports (i.e. a *global* RR)[4] has the potential to maximize the match size, whereas a longest queue first (LQF) scheduler can boost the match weight. Consider a global RR scheduler constructed based on the preferred input-output relationship in (1). In each time slot, N conflict-free input-output connections are naturally guaranteed – a big advantage for maximizing match size. But it is hard to ensure that all inputs will always have packets for their preferred outputs. Nevertheless, the global RR will give the ideal performance of 100% throughput and minimal packet delay under the extremely heavy and uniform traffic load. If the traffic is uniform *but* not heavy enough, 100% throughput can still be obtained but the delay will not be minimized due to the non-work-conserving scheduling nature of RR. If the traffic is non-uniform and sufficiently heavy, both 100% throughput and minimal delay performances are hard to achieve.

Next we consider an LQF scheduler, such as iLOF-1 [16]. Each input sends its VOQ sizes to all backlogged outputs as requests. Each output grants the request from the longest VOQ. The longest VOQ will be served with the highest priority. But the match size (after one iteration) tends to be small because unlike RR, multiple outputs may grant the same input. In general, it is impossible to guarantee 100% throughput unless the load is very light. When the traffic load is very light, most VOQs will be empty and LQF can allow (the few) backlogged VOQs to send packets quickly yet without affecting the throughput, because the match size is of less concern at light load. Notably, if the load is so light that there is only a single backlogged VOQ at an input, then this VOQ is also the longest VOQ of that input. Accordingly, random arbiters (as in PIM [15]) can give a comparable performance as LQF under light load. But when the traffic load is non-uniform and heavy, LQF-N outperforms random arbiters.

We can see that the global RR scheduler focuses on maximizing the match size by taking all input-output pairs into account, where VOQ occupancy is of less concern. On the other hand, LQF targets at addressing individual longest/critical VOQs, where the match size is of less concern. A good scheduler should try to get the best out of both, and

[4]This is in contrast to the port-based *local* RR schedulers/arbiters used by, e.g. iSLIP [10].

more importantly, in a way that can adapt to the real-time traffic fluctuation. With the above in mind, let us revisit the design philosophy of two representative algorithms iSLIP [10] and SRR [21].

B. Lessons Learned From iSLIP and SRR

iSLIP [10] was designed to address the poor high load performance of PIM [15] (with a small number of iterations). PIM uses $2N$ port-based local random arbiters. Its main problem is that local random arbiters fail to desynchronize the selections made by individual outputs/inputs at high and/or non-uniform traffic load. Counterintuitively, iSLIP tackles the problem by converting local random arbiters to *local* RR arbiters. At light load, local RR arbiters behave like random arbiters, and thus iSLIP can handle light traffic well. As load increases to medium, the performance of iSLIP (as well as PIM) is limited due to (a) its inability of identifying the longest VOQ (among requests received), and (b) its not-yet-fully exploited ability of desynchronizing grants issued by outputs. As load continues to increase, the joint efforts of $2N$ local RR arbiters gradually desynchronize requests/grants issued by inputs/outputs, resulting in a match size comparable to that of a global RR scheduler.

SRR [21] was designed to mimic a global RR scheduler using the preferred input-output relationship. Its strength is thus at uniform and heavy traffic load. To address the problem that some inputs have no packets for their preferred outputs, SRR allows an input to select its (local) longest VOQ to request instead. Note that a single-bit-request cannot carry the queue size information. When an output receives requests from multiple inputs, the preferred request (from the preferred input) will be granted; otherwise, a non-preferred request will be chosen randomly. We can see that the probability for a non-preferred request to be granted is rather small (i.e. only when no preferred request is received by the output and the particular non-preferred request is chosen by the output), and even if it is granted, the granted VOQ is unlikely the longest among all requests received. Further note that in the request phase, at most N requests will be sent. On average, each output will receive (no more than) one request. The choice for sending grant is very limited.

Overall, the performance of SRR is better than the global RR at light load, and the performance gap reduces when the load increases. SRR outperforms iSLIP-1 at high load because the match size and weight produced by SRR are generally larger than iSLIP-1.

From the above, we can see that both iSLIP and SRR have made efforts in getting the best out of the global RR and LQF schedulers, yet there is still room for improvement. Our new iteration scheduling algorithm is called Round Robin with Longest Queue First (RR/LQF). As the name implies, RR/LQF represents our efforts in getting the best out of RR and LQF.

C. RR/LQF

RR/LQF is a single-bit scheduling algorithm. It can run a single-iteration (RR/LQF-1), or multiple iterations (RR/LQF-N). In the request phase, each single-bit-request is an indication of a new packet arrival at the specific VOQ (instead of a matching request). As such, we rename the *request* phase to *report* phase. The single-bit-request becomes single-bit-report. When the input traffic is admissible, at most one packet can arrive at each input in each time slot. Our single-bit-reports allow each output to accurately keep track of the size of N VOQs destined to it. To be more specific, each output j maintains N packet counters C(i,j)s, one for each VOQ(i,j), where $i = 0, 1, \ldots, N-1$. Upon the arrival of single-bit-reports, each associated counter is increased by one. When a packet is scheduled for transmission, the associated counter will be decreased by one. Note that single-bit-report can also be used by other iterative scheduling algorithms such as iLQF [16] to minimize the communication overhead.

Like SRR [21], RR/LQF gives the scheduling priority to the preferred input-output pairs first. The subtle difference between RR/LQF and SRR is at handling the case that an input has no packet for its preferred output: RR/LQF always ensures that the longest VOQ is given the highest scheduling priority, whereas SRR cannot (see Section II.B). In the grant phase of RR/LQF, if the preferred input's VOQ is empty, output port j grants the (non-preferred) input that has the longest VOQ *among all VOQs destined to output* j. In the accept phase, an input i accepts the grant from its preferred output first. If there is no preferred grant, the grant for the longest VOQ at input i is accepted. The three-phase operation of RR/LQF at time slot t is summarized below.

Report: If there is a new packet arrival at VOQ(i,j) in slot t, input i sends a single-bit-report to output j.

Grant: If output j receives a report from input i, packet counter C(i,j) is increased by 1. Output j implicitly identifies its preferred input, say i, from (1). If C$(i,j) > 0$, output j sends a single-bit-grant to its preferred input i. Otherwise, output j grants the VOQ that has the largest C(i,j), where $i = 0, 1, \ldots, N-1$. (In case of a tie, the winner is selected by a local RR.)

Accept: Input i implicitly identifies its preferred output j from (1). If input i receives a grant from its preferred output j, a single-bit-accept is issued. Otherwise, among the grants received, the one corresponding to input i's longest VOQ is accepted. (In case of a tie, a local RR is adopted to select the winner.) When output j receives an accepting bit from input i, output j and input i are *matched*. Packet counter C(i,j) is then decreased by 1. The matched pairs will not participate in the subsequent scheduling iterations of the same time slot t.

Note that in the first/only iteration of RR/LQF-1 or the last iteration of RR/LQF-N, the accept phase is redundant. For RR/LQF-N, the first iteration carries out all three phases, and the remaining iterations only consist of the grant and accept phases. This is another advantage of using single-bit-report.

IV. Stability Proof of RR/LQF

In this section, we prove that RR/LQF only needs a speedup bound of 2-1/N to be stable, whereas all other iterative scheduling algorithms need a speedup bound of 2. Like the analytical efforts in [28] and [29], we adopt the fluid model to prove the stability under admissible traffic pattern.

A. Constructing a Fluid Model

We first construct a fluid model for RR/LQF. Let λ_{ij} be the mean packet arrival rate to VOQ(i,j), and $Z_{ij}(n)$ be the number of packets in VOQ(i,j) at the beginning of time slot n. Further let the cumulative number of packet arrivals and departures for VOQ(i,j) at the beginning of slot n be $A_{ij}(n)$ and $D_{ij}(n)$ respectively. We have:

$$Z_{ij}(n) = Z_{ij}(0) + A_{ij}(n) - D_{ij}(n), \quad n \geq 0,\ i,j = 0,\ldots,N-1. \tag{2}$$

Assume that the packet arrival process obeys the strong law of large numbers with probability one, i.e.

$$\lim_{n\to\infty} \frac{A_{ij}(n)}{n} = \lambda_{ij}, \quad i,j = 0,\ldots,N-1.$$

The switch is, by definition, rate stable if:

$$\lim_{n\to\infty} \frac{D_{ij}(n)}{n} = \lambda_{ij}, \quad i,j = 0,\ldots,N-1.$$

An admissible traffic matrix is defined as the one that satisfies the following constraints.

$$\sum_i \lambda_{i,j} \leq 1, \quad \sum_j \lambda_{i,j} \leq 1 \tag{3}$$

If a switch is rate stable for an admissible traffic matrix, then the switch delivers 100% throughput.

The fluid model is determined by a limiting procedure illustrated below. First, the discrete functions are extended to *right continuous* functions. For arbitrary time $t \in [n, n+1)$:

$$\begin{aligned} A_{ij}(t) &= A_{ij}(n); \\ Z_{ij}(t) &= Z_{ij}(n); \\ D_{ij}(t) &= D_{ij}(n) + (t-n)(D_{ij}(n+1) - D_{ij}(n)); \end{aligned}$$

Note that all functions are random elements of $\mathbb{D}[0, \infty)$. We shall sometimes use the notation $A_{ij}(\cdot,\omega)$, $Z_{ij}(\cdot,\omega)$ and $D_{ij}(\cdot,\omega)$ to explicitly denote the dependency on the sample path ω. For a fixed ω, at time t, we have [28]:

$A_{ij}(t,\omega)$, the cumulative number of arrivals to VOQ(i,j)

$Z_{ij}(t,\omega)$, the number of packets in VOQ(i,j)

$D_{ij}(t,\omega)$, the cumulative number of departures from VOQ(i,j)

For each $r > 0$, we define

$$\begin{aligned} \bar{A}^r_{ij}(t,\omega) &= r^{-1} A_{ij}(rt,\omega); \\ \bar{Z}^r_{ij}(t,\omega) &= r^{-1} Z_{ij}(rt,\omega); \\ \bar{D}^r_{ij}(t,\omega) &= r^{-1} D_{ij}(rt,\omega); \end{aligned}$$

It is shown in [29] that for each fixed ω satisfying (2) and any sequence $\{r_n\}$ with $r_n \to \infty$ as $n \to \infty$, there exists a subsequence $\{r_{n_k}\}$ and the continuous functions $(\bar{A}_{ij}(.), \bar{Z}_{ij}(.)\ldots)$, where $(\bar{A}^r_{ij}(t,\omega), \bar{Z}^r_{ij}(t,\omega)\ldots)$ converges to uniformly on compacts as $k \to \infty$ for any $t \geq 0$

$$\begin{aligned} \bar{A}^{r_{n_k}}_{ij}(t,\omega) &\to \lambda_{ij} t; \\ \bar{Z}^{r_{n_k}}_{ij}(t,\omega) &\to \bar{Z}_{ij}(t); \\ \bar{D}^{r_{n_k}}_{ij}(t,\omega) &\to \bar{D}_{ij}(t); \end{aligned} \tag{4}$$

Definition 1: Any function obtained through the limiting procedure in (4) is said to be a *fluid limit* of the switch. The fluid model equation using RR/LQF is:

$$\bar{Z}_{ij}(t) = \bar{Z}_{ij}(0) + \lambda_{ij} t - \bar{D}_{ij}(t) \quad t \geq 0 \tag{5}$$

Definition 2: The fluid model of a switch operating under a scheduling algorithm is said to be weakly stable if for every fluid model solution $(\bar{D}, \bar{Z})$ with $\bar{Z}(0) = 0$, $\bar{Z}(t) = 0$ for almost every $t \geq 0$.

B. 100% Throughput Proof

For any input port i, let $\lambda_{ij} = \max\{\lambda_{im}\}$, where $m = 0, 1 \ldots N-1$. For any given admissible traffic matrix, by conditioning on λ_{ij}, two possible cases are $0 \leq \lambda_{ij} < 1/N$ and $1/N \leq \lambda_{ij} \leq 1$.

Statement 1: If $1/N \leq \lambda_{ij} \leq 1$, RR/LQF requires a speedup of 2-$1/N$ to guarantee 100% throughput.

Proof: Let $C_{ij}(t)$ denote the joint queue occupancy of all packets arrived at input port i, plus all packets destined for output port j. Since flow(i,j) is counted twice by input i and output j respectively, we subtract it once from the joint occupancy:

$$C_{ij}(t) = \sum_p \bar{Z}_{ip}(t) + \sum_m \bar{Z}_{mj}(t) - \bar{Z}_{ij}(t) \tag{6}$$

$\bar{Z}(t)$ is a non-negative, absolutely continuous function, so $C_{ij}(t)$ is also non-negative and absolutely continuous from (6). Without loss of generality, assume all VOQs are initially empty, i.e. $\bar{Z}(0) = 0$. Then $C_{ij}(0) = 0$ and the derivative of $C_{ij}(t)$ is:

$$C'_{ij}(t) = \sum_p \bar{Z}'_{ip}(t) + \sum_m \bar{Z}'_{mj}(t) - \bar{Z}'_{ij}(t)$$

Combine the above equation with (5), and we get

$$C'_{ij}(t) = \sum_p \lambda_{ip} + \sum_m \lambda_{mj} - \lambda_{ij} - [\sum_p \bar{D}_{ip}(t) + \sum_m \bar{D}_{mj}(t) - \bar{D}_{ij}(t)]'$$

From the admissible traffic condition and $1/N \leq \lambda_{ij} \leq 1$,

$$C'_{ij}(t) \leq 2 - 1/N - [\sum_p \bar{D}_{ip}(t) + \sum_m \bar{D}_{mj}(t) - \bar{D}_{ij}(t)]' \tag{7}$$

Suppose that VOQ(i,j) is non-empty at time $t > 0$, i.e. $\bar{Z}_{ij}(t) > 0$, then by the continuity of $\bar{Z}(t)$, $\exists\delta$ such that $\bar{Z}_{ij}(t') > 0$ for $t' \in [t, t+\delta]$. Let

$$a = \min_{t' \in [t,t+\delta]} \bar{Z}_{ij}(t')$$

For a large enough k, we have $\bar{Z}^{r_{n_k}}_{ij}(t') \geq a/2$, where $t' \in [t, t+\delta]$. Also for a large enough k, we have $r_{n_k} \cdot a/2 \geq 1$. Thus $Z_{ij}(t') \geq 1$ where $t' \in [r_{n_k}t, r_{n_k}(t+\delta)]$. This means that VOQ$(i,j)$ holds at least one packet in the long time interval $[r_{n_k}t, r_{n_k}(t+\delta)]$. Thanks to the MSM, during the same interval, input i sends or output j receives a packet per

time slot. If the switch is operated with a speedup of S, in a long time interval $[r_{n_k}t,r_{n_k}(t+\delta)]$,$t' \in [t,t+\delta]$ it fulfills:

$$\sum_p \left[D_{ip}\left(r_{n_k}t'\right) - D_{ip}\left(r_{n_k}t\right)\right] + \sum_m \left[D_{mj}\left(r_{n_k}t'\right) - D_{mj}\left(r_{n_k}t\right)\right] - \left[D_{ij}\left(r_{n_k}t'\right) - D_{ij}\left(r_{n_k}t\right)\right] \geq S \cdot r_{n_k}(t'-t)$$

Dividing the above equation with r_{n_k} and letting $k \to \infty$, fluid limit is obtained as:

$$\sum_p \left[\bar{D}_{ip}\left(t'\right) - \bar{D}_{ip}\left(t\right)\right] + \sum_m \left[\bar{D}_{mj}\left(t'\right) - \bar{D}_{mj}\left(t\right)\right] - \left[\bar{D}_{ij}\left(t'\right) - \bar{D}_{ij}\left(t\right)\right] \geq S \cdot (t'-t)$$

Further dividing the above equation by $(t'-t)$ and letting $t' \to t$, the derivative of fluid limit is

$$[\sum_p \bar{D}_{ip}\left(t\right) + \sum_m \bar{D}_{mj}\left(t\right) - \bar{D}_{ij}\left(t\right)]' \geq S \tag{8}$$

With a speedup of $S = 2 - 1/N$, combining (7) and (8), we get

$$C'_{ij}\left(t\right) \leq 0$$

We borrow the following **Fact 1** from [28]:

Fact 1: Let f be a non-negative, absolutely continuous function defined on $\mathbf{R}^+ \cup \{0\}$ with $f(0) = 0$. Assume that for almost every t such that $f\left(t\right) > 0, f'(t) \leq 0$. Then $f(t) = 0$ for almost every $t \geq 0$.

Based on **Fact 1**, $C_{ij}(t) = 0$ for almost every $t \geq 0$. Due to (6) and $C_{ij}(t) = 0$, $\bar{Z}_{ij}\left(t\right) = 0$ for almost every $t \geq 0$. We can see that RR/LQF is weakly stable (**Definition 2**). From [29], the switch is rate stable if its corresponding fluid model is weakly stable. Then we proved **Statement 1** that when $1/N \leq \lambda_{ij} \leq 1$, RR/LQF requires a speedup of 2-1/N to guarantee 100% throughput. #

Statement 2: When $0 \leq \lambda_{ij} < 1/N$, RR/LQF is stable without any speedup.

Proof: Define $\mathbf{B} \triangleq \{m : \bar{Z}_{im}\left(t\right) > 0\}$. Let $G_i(t)$ denote the joint queue occupancy of all non-empty VOQs at input port i:

$$G_i\left(t\right) = \sum_{m \in B} \bar{Z}_{im}\left(t\right) \tag{9}$$

$\bar{Z}\left(t\right)$ is a non-negative, absolutely continuous function, so $G_i(t)$ is also non-negative and absolutely continuous from (9). Without loss of generality, assume all VOQs are initially empty, i.e. $\bar{Z}\left(0\right) = 0$. Then $G_i(0) = 0$ and the derivative of $G_i(t)$ is

$$G'_i\left(t\right) = \sum_{m \in B} \bar{Z}'_{im}\left(t\right)$$

Combine the above equation with (5), and we get

$$G'_i\left(t\right) = \sum_{m \in B} \lambda_{im} - \sum_{m \in B} \bar{D}'_{im}\left(t\right)$$

From the conditions $\lambda_{ij} = \max\{\lambda_{im}\}(m = 0, 1, \ldots, N-1)$ and $0 \leq \lambda_{ij} < 1/N$,

$$G'_i\left(t\right) < \frac{h}{N} - \sum_{m \in B} \bar{D}'_{im}\left(t\right) \tag{10}$$

where $h = \|\mathbf{B}\| \geq 0$.

Suppose that $G_i(t) > 0$ when $t > 0$. This implies that for $\forall m_1 \in \mathbf{B}$ and $\forall m_2 \notin \mathbf{B}$, $\bar{Z}_{im_1}\left(t\right) > 0$ and $\bar{Z}_{im_2}\left(t\right) = 0$. Then $\bar{Z}_{im_1}\left(t\right) - \bar{Z}_{im_2}\left(t\right) > 0$. By the continuity of these functions, $\exists \delta$ such that

$$\min_{t' \in\ [t,t+\delta]} \bar{Z}_{im_1}\left(t'\right) - \bar{Z}_{im_2}\left(t'\right) > 0, \quad \text{for} \quad \forall m_1 \in \mathbf{B} \quad \text{and} \quad \forall m_2 \notin \mathbf{B}$$

Let

$$q = \min_{\substack{m'_1 \in \mathbf{B} \\ m'_2 \notin \mathbf{B}}} \min_{t' \in\ [t,t+\delta]} \{\bar{Z}_{im_1}\left(t'\right) - \bar{Z}_{im_2}\left(t'\right)\}$$

Thus for a large enough k, we have $\bar{Z}^{\tau_{n_k}}_{im_1}\left(t'\right) - \bar{Z}^{\tau_{n_k}}_{im_2}\left(t'\right) \geq q/2$, where $\forall m_1 \in \mathbf{B}$, $\forall m_2 \notin \mathbf{B}$, and $t' \in [t,t+\delta]$. Also for a large enough k, we have $r_{n_k} \cdot q/2 \geq 1$. Thus $Z_{im_1}\left(t'\right) - Z_{im_2}\left(t'\right) \geq 1$, where $\forall\ m_1 \in \mathbf{B}$, $\forall m_2 \notin \mathbf{B}$, and $t' \in [r_{n_k}t,r_{n_k}(t+\delta)]$. This means that in the long time interval $[r_{n_k}t,r_{n_k}(t+\delta)]$, any non-empty VOQ at input port i belongs to the set $\mathbf{U} \triangleq \{VOQ(i,m)\text{: } m \in \mathbf{B}\}$, and any VOQ that belongs to set $\mathbf{U}$ is non-empty [28], [29]. Since RR/LQF always gives the highest priority to the preferred input-output pairs calculated by (1), during the same interval, each non-empty VOQ sends at least one packet per N time slots. Then in the long time interval $[r_{n_k}t,r_{n_k}(t+\delta)]$, input i sends at least $h = \|\mathrm{B}\|$ packets per N slots,

$$\sum_{m \in B} \left[D_{im}\left(r_{n_k}t'\right) - D_{im}\left(r_{n_k}t\right)\right] \geq Lh \tag{11}$$

where $L \in \mathbb{Z}$, $NL \leq r_{n_k}t' - r_{n_k}t < NL + N$. So we have

$$L > \frac{r_{n_k} \cdot (t'-t)}{N} - 1 \tag{12}$$

Combine (11) with (12),

$$\sum_{m \in \mathrm{B}} \left[D_{im}\left(r_{n_k}t'\right) - D_{im}\left(r_{n_k}t\right)\right] > \frac{h \cdot r_{n_k} \cdot (t'-t)}{N} - h$$

Since $h = \|\mathrm{B}\|$ is within [0, N], its impact is insignificant for fluid limit [28]. Dividing the above equation with r_{n_k} and letting $k \to \infty$, the fluid limit is obtained as:

$$\sum_{m \in \mathrm{B}} \left[\bar{D}_{im}\left(t'\right) - \bar{D}_{im}\left(t\right)\right] > \frac{h \cdot (t'-t)}{N}$$

Further dividing the above equation by $(t'-t)$ and letting $t' \to t$, the derivative of fluid limit is

$$\sum_{m \in \mathrm{B}} \bar{D}'_{im}\left(t\right) > \frac{h}{N} \tag{13}$$

Combine (10) and (13), and then we get

$$G'_i\left(t\right) < 0$$

Based on **Fact 1**, $G_i(t) = 0$ for almost every $t \geq 0$. Due to (9) and $G_i(t) = 0$, $\bar{Z}_{im}\left(t\right) = 0$ for almost every $t \geq 0$.

Then RR/LQF is weakly stable. We proved **Statement 2** that if $0 \leq \lambda_{ij} < 1/N$, RR/LQF is stable without any speedup. #

Combining **Statements** 1 and 2 above, we have:

Theorem 1 (Sufficiency): RR/LQF can achieve 100% throughput with a speedup of 2-1/N for any admissible traffic pattern obeying the strong law of large numbers.

C. Generalization of RR/LQF

The proof for **Statements 1** and **2** is based on the properties of MSM (Maximal Size Match) and global RR (i.e. the preferred input-output pairs) in RR/LQF. In other words, MSM and global RR jointly tighten the speedup bound to 2-1/N. For iterative algorithms (e.g. iSLIP [10]) that do not have the global RR in their design, the best/tightest speedup bound obtained is 2. We can see that the additional 1/N reduction in speedup comes from the global RR. In essence, the global RR ensures that each non-empty VOQ sends at least one packet per N time slots, whereas iSLIP and alike can only guarantee one packet per N^2 slots. From this insight, **Theorem 1** can be generalized to **Theorem 2**.

Theorem 2 (Sufficiency): As long as an iterative algorithm guarantees the global RR and MSM after N iterations, it can achieve 100% throughput with a speedup of 2-1/N for any admissible traffic pattern obeying the strong law of large numbers.

We omit the proof for **Theorem 2** because it is similar to that of **Theorem 1**. Accordingly, we can have a new class of iterative algorithms RR/* (where * is the arbitration for non-preferred VOQs). For example, let each input/output port select the non-preferred VOQ using a local RR pointer/arbiter like that in iSLIP. The resulting RR/RR algorithm is stable with the same speedup bound of 2-1/N. Compared with RR/LQF, RR/RR is simpler to implement (see Section VII.C) and yields better fairness performance (see Appendix). But its delay-throughput performance is not as good as RR/LQF.

V. Performance Evaluations

In this section, we study the performance of RR/LQF by simulations. We compare it with the following representative iterative scheduling algorithms, PIM [15], iSLIP [10], iOCF [16], iLQF [16] and SRR [21]. The number of iterations executed by an algorithm varies. We use, e.g. iSLIP-2 to denote iSLIP with 2 iterations. Note that SRR is designed for single iteration only. For clarity, we use SRR-1 to identify it in this section. The performance of an output-queued switch is also simulated to provide the lower bound on performance. Besides, no switch speedup is implemented in out simulations. Three classic traffic patterns [10], [16] are considered, uniform, bursty and hot-spot. In the following, we only present simulation results for switch with size $N = 32$, yet the same conclusions apply to other sizes. Finally, each point of data in Figs. 2–7 is based on a simulation run of 10^5 time slots with the initial 5×10^4 slots as "warm-up".

A. Uniform Traffic

At every time slot for each input, a packet arrives with probability p (input load p) and destines to each output with equal probability. In Fig. 2, we can see that RR/LQF-2 obtains 100% throughput and the best delay performance among all iterative scheduling algorithms. iOCF-2 and iLQF-2 only achieve less than 90% throughput. Even under a single-iteration, RR/LQF-1 outperforms SRR-1. Compared with iSLIP-2, RR/LQF-2 gives much smaller delay. When $p = 0.9$, iSLIP-2 requires 38 time slots and RR/LQF-2 only 18, cutting down the delay by more than 50%. This is mainly because RR/LQF combines and maximizes the advantages of RR and LQF.

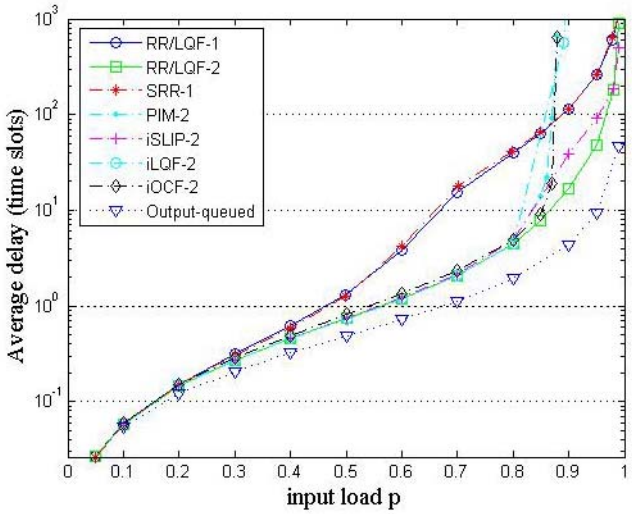


Fig. 2. Delay vs input load, under uniform traffic.

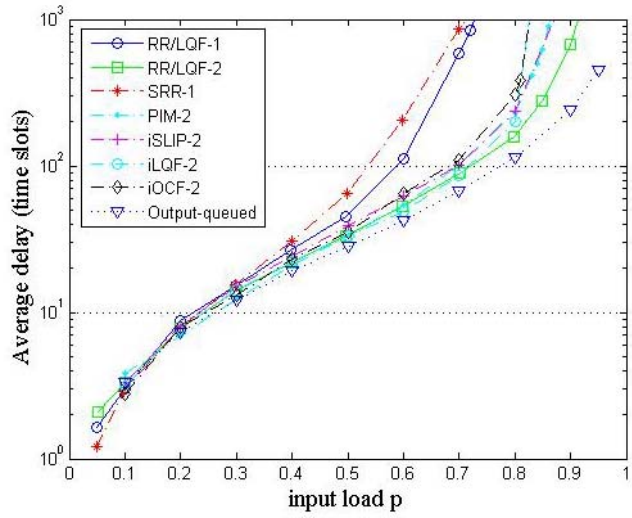


Fig. 3. Delay vs input load, under bursty traffic.

B. Bursty Traffic

Bursty arrivals are modeled by the ON/OFF traffic model, which is a special instance of the two-state Markov-modulated process. 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 burst size w, the state transition probabilities from OFF to ON is $p/[w(1-p)]$ and from ON to OFF is $1/w$. We set the burst size $w = 30$ packets.

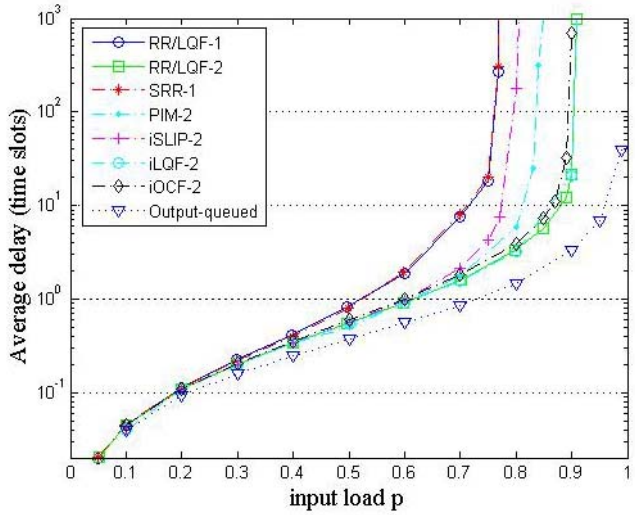


Fig. 4. Delay vs input load, under hot-spot traffic.

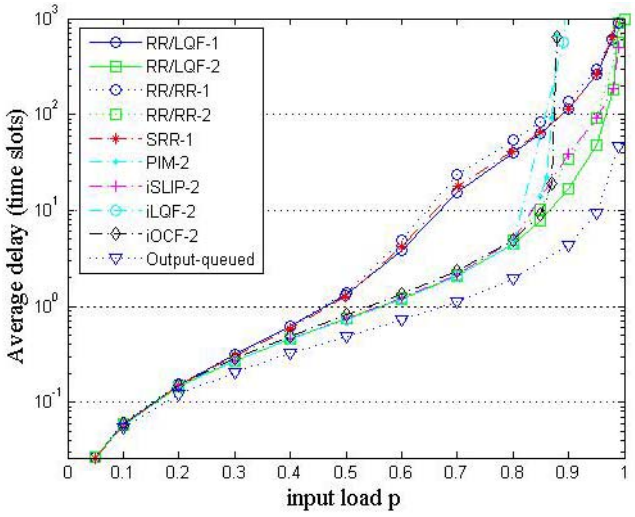


Fig. 6. RR/RR vs other algorithms under uniform traffic.

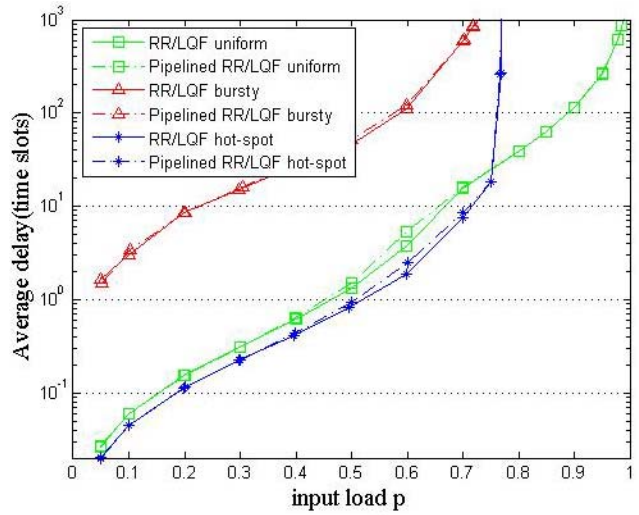


Fig. 5. Delay vs input load, pipelined RR/LQF and RR/LQF.

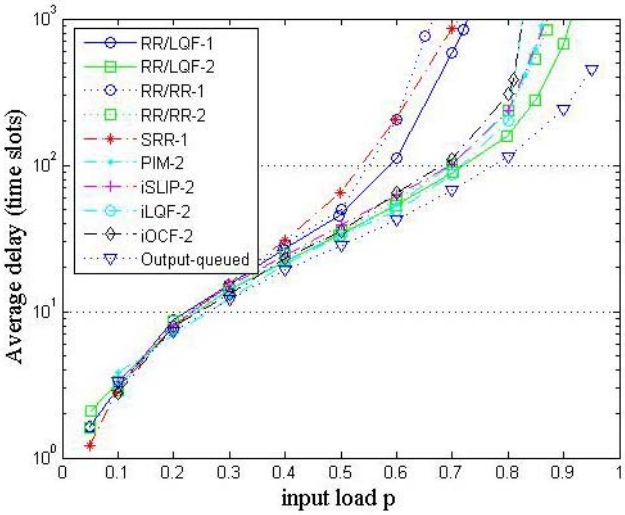


Fig. 7. RR/RR vs other algorithms under bursty traffic.

From Fig. 3, we can see that delay builds up quickly with input load. Among all iterative scheduling algorithms, RR/LQF-2 still gives the lowest/best delay performance. In terms of throughput, iSLIP-2 and iLQF-2 only achieve less than 85% but RR/LQF-2 obtains more than 90%. RR/LQF-1 also provides a much higher throughput than SRR-1. At $p = 0.8$, iSLIP-2 packets experience an average delay of 238 time slots, whereas for RR/LQF-2 just 167, cutting down the delay by almost 30%.

C. 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. 4, again RR/LQF-2 outperforms iSLIP-2 and iLQF-2. RR/LQF-1 yields the lower delay than SRR-1. As an example, with a delay requirement of no more than 30 slots, RR/LQF-2 can carry a load of $p = 0.9$ whereas iSLIP-2 can only carry 0.7. A material benefit is that the load-carrying capacity is increased by more than 20%.

In summary, despite the fact that RR/LQF requires the least communication overhead (i.e. a single bit for request, grant and accept phases), it shows the best delay/throughput performance under all three classic traffic patterns.

VI. Fairness of RR/LQF

Under admissible traffic pattern, 100% throughput can guarantee all packets are delivered within a bounded delay. Therefore, the fairness in bandwidth allocation is not an issue. In practice, the incoming traffic is likely to be inadmissible (especially at shorter time scale), where an output port could be oversubscribed (i.e. $\sum_i \lambda_{i,j} > 1$). In this case, the fairness for bandwidth allocation is essential. Among various fairness criteria [30], the max-min fairness (see **Definition 5**) is widely adopted.

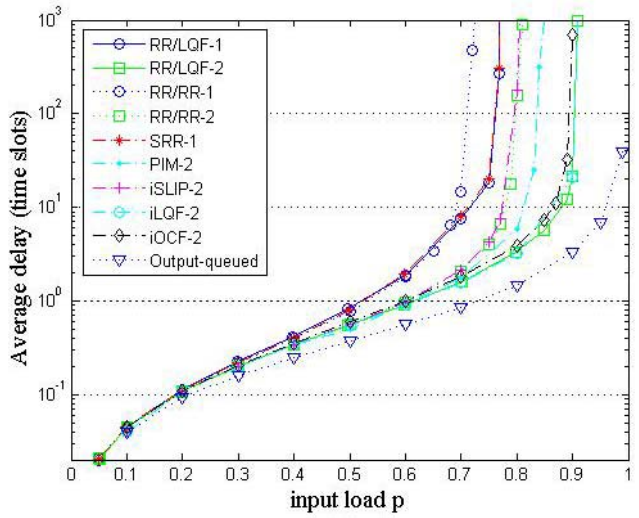


Fig. 8. RR/RR vs other algorithms under hot-spot traffic.

In an $N \times N$ input-queued switch, the service demand is the input traffic load. The service allocation is the bandwidth actually received. For each flow(i, j), we let λ_{ij} and a_{ij} be its traffic demand and bandwidth allocation respectively. Then the $N \times N$ matrix $A = \{a_{ij}\}$, $i = 1, \ldots, N, j = 1, \ldots, N$ is the allocation result for whole switch. We also use U_i and V_j to denote the capacity of input i and output j respectively. Three definitions from [30] are borrowed here.

Definition 3: A flow flow(i, j) is said to be satisfied if its allocation is equal to its demand, that is, $a_{ij} = \lambda_{ij}$. Otherwise, the flow(i, j) is said to be unsatisfied.

Definition 4: The allocation matrix A is said to be feasible if and only if:

1) Each flow receives an allocation greater than or equal to zero; that is, for all i and j, $a_{ij} \geq 0$.
2) The total allocation of each input or output is no more than the capacity of that input or output; that is,

$$\sum_{\forall j} a_{ij} \leq U_i, \quad \sum_{\forall i} a_{ij} \leq V_j \tag{14}$$

Definition 5: The allocation matrix A is said to be max-min fair if and only if:

1) It is feasible.
2) No flow receives an allocation greater than its demand; that is, for all i and j,$a_{ij} \leq \lambda_{ij}$.
3) For all i and j, the allocation a_{ij} of flow(i, j) cannot be increased while satisfying the above two conditions and without reducing the allocation $a_{xj}(x = 1 \ldots N, x \neq i)$ or a_{iy} $(y = 1 \ldots N, y \neq j)$, where $a_{ij} \geq a_{xj}$ and $a_{ij} \geq a_{iy}$.

As long as an algorithm meets the three conditions above, it satisfies the max-min fair criterion. RR/LQF can guarantee the max-min fairness under the following traffic conditions.

1) λ_{ij} is identical, for all $i = 1, \ldots, N, j = 1, \ldots, N$. All flows have the same demand λ_{ij}. If $\lambda_{ij} < 1/N$, all flows are satisfied. Otherwise, all flows are unsatisfied but receive the same allocation $1/N$ due to the global RR. In any case, RR/LQF is max-min fair.

2) $\lambda_{ij} \leq 1/N$, for all $i = 1, \ldots, N, j = 1, \ldots, N$. The demand of each flow is no larger than $1/N$. The global RR guarantees the minimum allocation $1/N$ for each flow. All flows are satisfied. RR/LQF is max-min fair.

3) $\lambda_{ij} > 1/N$, for all $i = 1, \ldots, N, j = 1, \ldots, N$. The demand of each flow is larger than $1/N$. All flows are unsatisfied but receive the same allocation $1/N$ due to the global RR. RR/LQF is max-min fair.

4) $\lambda_{ij} > 1/N$, $\lambda_{xj} \leq 1/N$ $(x = 1 \ldots N, x \neq i)$ and $\lambda_{iy} \leq 1/N$ $(y = 1 \ldots N, y \neq j)$, for all $i = 1, \ldots, N, j = 1, \ldots, N$. This is the case that only one flow at input i and output j, i.e. flow(i, j), has a demand larger than $1/N$. In this case, the use of RR scheduler ensures that all flows (at input i and output j) except flow(i, j) are satisfied. Flow(i, j) will take all residual bandwidth (i.e. $1/N - \lambda_{iy}$ and $1/N - \lambda_{xj}$) from other flows due to the use of LQF scheduler. If flow(i, j) is also satisfied, then all flows are satisfied. RR/LQF is thus max-min fair. Otherwise, VOQ(i, j) will be always backlogged. Since RR/LQF is work-conserving, the always-backlogged VOQ(i, j) affirms that input i or output j has no unallocated bandwidth left. In order to increase the allocation of flow(i, j), we have to reduce the allocation of other flows at input i or output j (condition 3 of **Definition 5**). RR/LQF is then still max-min fair.

We can see that RR/LQF satisfies the max-min fair criterion under many traffic patterns, but it will still fail for some cases. Consider the case that $\lambda_{ij} \leq 1/N$ for all except two flows at input i or output j. The global RR tries to allocate each flow with an equal bandwidth of $1/N$. Each flow(i, j) with demand $\leq 1/N$ is satisfied and has an excess of $1/N - \lambda_{ij}$. The two flows with demand $> 1/N$ cannot be satisfied. But they can use the excess bandwidth from the satisfied flows using LQF. Notably, LQF will allow the flow with a higher demand to have a bigger slice of the excess bandwidth. As a result, RR/LQF cannot ensure max-min fair bandwidth allocation between these two flows. In Appendix, we prove that the simpler variant of RR/LQF, RR/RR, guarantees the max-min fairness under any traffic scenario.

VII. Implementation Issues

A. Out-of-Syn Problem

The simplicity of single-bit-report phase may come with a cost when the states of VOQs at the inputs and their counters maintained by the outputs (arbiters) get out-of-sync. To address this issue, a simple synchronization mechanism is designed as follows. In the accept phase of RR/LQF, input i implicitly identifies its preferred output j from (1). If VOQ(i, j) is not empty but does not receive a grant from output j (or vice versa), input i can detect the out-of-sync between its VOQ state and that of output j. This implicit detection for each VOQ is carried out once in every N time slots. It is deemed sufficient as out-of-sync events should be rare. When an out-of-sync is indeed detected, input i can send a special control packet to inform output j its correct queue size. We assume such control overhead is minimal, and thus not considered in our simulations.

B. Pipelined RR/LQF

To implement RR/LQF, the complexity is dominated by the computation overhead in identifying the longest VOQ. Recall that each output j (arbiter) maintains N packet counters C(i, j), $(i = 0, 1, \ldots, N-1)$, denoting the size of each VOQ(i, j) destined to it. In order to identify the longest VOQ, counters (i.e. VOQ sizes) are ranked such that the largest counter ranked 1st and smallest counter ranked N-th. The ranking list is then stored in a balanced binary search tree [31]. In each time slot, an output can receive at most one accept message. When an accept from input i arrives at output j, counter C(i, j) is decreased by one (indicating a packet is to be sent). Since other counters at output j remain unchanged, the rank of C(i, j) will be increased. The time complexity for finding the new rank of C(i, j) in a binary tree is $O(\log N)$.

In RR/LQF, an output j (arbiter) may receive multiple (up to N) single-bit-reports at a specific time slot t. All associated packet counters C(i, j)s will be increased by one immediately (as such update effort is minimal). Output j is required to re-rank all updated counters in slot t, and the worst-case complexity is $O(N \log N)$. This complexity may be a potential bottleneck for high-speed implementation. To address this issue, we propose a pipelined re-ranking scheme, or pipelined RR/LQF. The pipelined RR/LQF has a complexity of $O(\log N)$ because only one counter is re-ranked in each time slot.

In the pipelined RR/LQF, a counter C(i, j) is *pending* if its value has been increased (due to the arrival of a single-bit-report) but its rank has *not* been updated yet. When output j receives some single-bit-reports at slot t, the associated counters are increased by one and immediately become pending. Among all counters becoming pending at slot t (plus pending counters from previous slots), output j selects one, say C(i, j), for updating its rank in slot t. As soon as the rank of C(i, j) is updated, its pending status is removed. Note that while a pending counter is waiting for its turn to be updated, its counter value is increased as usual for each new single-bit-report arrival. Besides, all pending counters will be updated in at most N slots. This is ensured by the simple counter selection mechanism: If the counter for the preferred VOQ is pending, it is selected; otherwise, the counter with the highest rank among all pending counters is selected. From (1), each VOQ is preferred once every N time slots. Therefore, all pending counters can have their ranks updated within N slots. The pipelined RR/LQF is thus stable.

We compare the performance of pipelined RR/LQF with RR/LQF by simulations in Fig. 5 under uniform, bursty and hot-spot traffic models. It can be seen that the pipelined RR/LQF is comparable to RR/LQF under almost all traffic loads. This shows that the pipelined re-ranking of pending counters has minimal effect on the delay-throughput performance.

C. RR/RR

The pipelined RR/LQF cuts down the computation complexity from $O(N \log N)$ to $O(\log N)$. To further reduce the complexity, RR/RR can be adopted. Like RR/LQF, RR/RR also gives the highest priority to the preferred input-output pairs in (1). When the preferred pair (i.e. global RR) fails, RR/RR uses a local port-based RR arbiter in selecting a VOQ to serve (instead of the original longest queue in RR/LQF). As a result, the complexity for sorting/ranking counters is eliminated. In essence, each port only implements a local RR arbiter with complexity $O(1)$. Note that the local RR pointer of each port does *not* move if the preferred VOQ is matched for transmission. Only after the non-preferred VOQ(i, j) sends a packet, the local RR pointers of input i and output j are updated to the next of VOQ(i, j).

In Appendix, we prove that RR/RR satisfies the max-min fair criterion under any traffic pattern. From the simulation results in Appendix, RR/RR yields similar performance as RR/LQF under uniform traffic. Under non-uniform traffic, more significant performance degradation is observed but still better than most existing algorithms.

VIII. Conclusions

In this paper, a new iterative scheduling algorithm called Round Robin with Longest Queue First (RR/LQF) was proposed. Notably, RR focuses on maximizing the match size and LQF targets at addressing the most critical VOQs. RR/LQF gets the best out of both. RR/LQF is a single-bit scheduling algorithm. If a single iteration is executed in each time slot, we showed that RR/LQF-1 gives the best delay-throughput performance among all single-bit-single-iteration scheduling algorithms. If up to N iterations are executed for finding the maximal size match in each time slot, we showed that RR/LQF-N is the first stable iterative scheduling algorithm with a speedup less than 2. We then generalized RR/LQF to become a class of algorithms that all have the some speedup bound of 2-1/N. To further cut down the implementation cost, we proposed the pipelined RR/LQF and RR/RR, a simpler variant of RR/LQF.

Appendix

In Appendix, we prove that RR/RR (proposed in Section VII.C) satisfies the max-min fair criterion under any traffic scenario. In an input-queued switch, the minimum bandwidth allocation is 0, so we can get $a_{ij} \geq 0$ for all $i = 1, \ldots, N, j = 1, \ldots, N$ (condition 1 of **Definition 4**). In each time slot, at most one packet is allowed to be sent/received by each input/output port. Then the equation (14) is true (condition 2 of **Definition 4**). From **Definition 4**, RR/RR is thus feasible. As long as a VOQ is empty (no matter it is preferred or not), no grant is issued for this VOQ and then it would not be matched for sure. Therefore, no allocation is wasted in RR/RR and $a_{ij} \leq \lambda_{ij}$ can be ensured for all $i = 1, \ldots, N, j = 1, \ldots, N$(condition 2 of **Definition 5**).

In the following, we focus on condition 3 of **Definition 5**, where we increase some bandwidth allocation a_{ij} and see how this would affect other flows. Conditioning on the flow(i, j) satisfied or not, two cases are:

- Flow(i, j) is satisfied. From **Definition 3**, $a_{ij} = \lambda_{ij}$. Then a_{ij} cannot be further increased due to constraint $a_{ij} \leq \lambda_{ij}$ in condition 2 of **Definition 5**.

- Flow(i, j) is unsatisfied. The VOQ(i, j) is always backlogged. Since RR/RR is a work-conserving algorithm, the unsatisfied flow(i, j) affirms that input i or output j has no unallocated bandwidth left,

$$\sum\nolimits_{\forall j} a_{ij} = U_i \text{ or } \sum\nolimits_{\forall i} a_{ij} = V_j \tag{15}$$

In RR/RR, the global RR allows each non-empty VOQ to deliver one packet for N time slots. The local port RR scheduler skips the empty VOQs and selects the next non-empty VOQ. Therefore, the always backlogged VOQ(i, j) would never waste any chance for sending packets by both the global RR and local RR turns. Then other VOQs at input i or output j would not get the larger allocation than that of the always backlogged VOQ(i, j):

$$\begin{aligned} &a_{ij} \geq a_{xj} \, (x = 1 \ldots N, x \neq i) \text{ or} \\ &a_{ij} \geq a_{iy} \, (y = 1 \ldots N, y \neq j) \end{aligned} \tag{16}$$

To increase a_{ij}, we have to reduce some a_{xj} $(x = 1 \ldots N, x \neq i)$ or a_{iy} $(y = 1 \ldots N, y \neq j)$ due to (15). From (16), we effectively reduce the allocation less than or equal to a_{ij} (condition 3 of **Definition 5**).

Combining the proofs for all three conditions [30] in **Definition 5**, RR/RR satisfies the max-min fair criterion in despite of traffic matrices.

We also collect the simulation results of RR/RR under three classic traffic patterns, uniform, bursty and hot-spot. From Fig. 6, RR/RR achieves 100% throughput and almost the same performance as RR/LQF under uniform traffic. For bursty (Fig. 7) and hot-spot (Fig. 8) traffic, RR/RR and RR/LQF are comparable when input load $p \leq 0.5$. This is because LQF does not show its superiority at low queue occupancy. Their performance gaps are bigger under non-uniform traffic with $p > 0.5$. This is the price paid for lower computation complexity.

ACKNOWLEDGMENT

The authors would like to thank the editor Prof. Marco Mellia and four anonymous reviewers for their detailed and insightful comments, which helped to significantly improve the quality of the paper.

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] J. L. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato, "Progressive congestion management based on packet marking and validation techniques," *IEEE Trans. Comput.*, vol. 61, no. 9, pp. 1293–1310, Sep. 2012.

[4] D. R. Hanks, Jr., *Juniper QFX5100 Series: A Comprehensive Guide to Building Next-Generation Networks*. Sebastopol, CA, USA: O'Reilly Media, Nov. 2014.

[5] C. Gauthier. (Nov. 2011). Overcoming 40G/100G SerDes design and implementation challenges. EE Times. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1279194

[6] Broadcom. *High-capacity Strataxgs Trident II 104×10G/32×40G Ethernet Multilayer Switch Family Featuring Warpcore Serdes Interfaces*, accessed on Mar. 19, 2016. [Online]. Available: https://www.broadcom.com/collateral/pb/56850-PB03-R.pdf

[7] "Getting started with Cisco nexus 9000 series switches in the small-to-midsize commercial data center guide," Cisco, San Jose, CA, USA, White Paper C07-733228-00, Sep. 2015. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/guide-c07-733228.html

[8] *High-Density 25/100 Gigabit Ethernet StrataXGS Tomahawk Ethernet Switch Series*, accessed on Mar. 19, 2016. [Online]. Available: https://www.broadcom.com/products/ethernet-communication-and-switching/switching/bcm56960-series

[9] "Cisco Nexus 5000 Series architecture: The building blocks of the unified fabric," Cisco, San Jose, CA, USA, White Paper C11-462176-03, Jun. 2009. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-462176.html

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

[11] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, no. 12, pp. 1347–1356, Dec. 1987.

[12] Y. Tamir and G. L. Frazier, "High-performance multi-queue buffers for VLSI communications switches," in *Proc. 15th Annu. Symp. Comput. Archit.*, Jun. 1988, pp. 343–354.

[13] G. Chartrand, *Introductory Graph Theory*. New York, NY, USA: Dover, 1985, p. 116.

[14] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. 15th IEEE INFOCOM*, San Francisco, CA, USA, Mar. 1996, pp. 296–302.

[15] T. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, 1993.

[16] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, 1995.

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

[18] G. Damm, J. Blanton, P. Golla, D. Verchère, and M. Yang, "Fast scheduler solutions to the problem of priorities for polarized data traffic," in *Proc. Int. Symp. Telecommun. (IST)*, Tehran, Iran, 2001, pp. 1–4.

[19] E. Leonardi, M. Mellia, M. A. Marsan, and F. Neri, "Stability of maximal size matching scheduling in input-queued cell switches," in *Proc. IEEE ICC*, New Orleans, LA, USA, Jun. 2000, pp. 1758–1763.

[20] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "On the stability of input-queued switches with speed-up," *IEEE/ACM Trans. Netw.*, vol. 9, no. 1, pp. 104–118, Feb. 2001.

[21] A. Scicchitano, A. Bianco, P. Giaccone, E. Leonardi, and E. Schiattarella, "Distributed scheduling in input queued switches," in *Proc. IEEE ICC*, Glasgow, Scotland, Jun. 2007, pp. 6330–6335.

[22] 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, Feb. 2008.

[23] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced Birkhoff–von Neumann switches, part I: One-stage buffering," *Comput. Commun.*, vol. 25, no. 6, pp. 611–622, 2002.

[24] B. Hu and K. L. Yeung, "Feedback-based scheduling for load-balanced two-stage switches," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1077–1090, Aug. 2010.

[25] B. Hu and K. L. Yeung, "Load-balanced optical switch for high-speed router design," *J. Lightw. Technol.*, vol. 28, no. 13, pp. 1969–1977, Jul. 1, 2010.

[26] K. Xi, Y.-H. Kao, and H. J. Chao, "A petabit bufferless optical switch for data center networks," in *Optical Interconnects for Future Data Center Networks*. New York, NY, USA: Springer, 2013, pp. 135–154.

[27] D. Lin, Y. Jiang, and M. Hamdi, "Selective-request round-robin scheduling for VOQ packet switch architecture," in *Proc. IEEE ICC*, Kyoto, Japan, Jun. 2011, pp. 1–5.

[28] M. S. Berger, "Delivering 100% throughput in a buffered crossbar with round robin scheduling," in *Proc. IEEE HPSR*, Poznań, Poland, Jun. 2006, pp. 403–407.

[29] T. Javidi, R. Magill, and T. Hrabik, "A high-throughput scheduling algorithm for a buffered crossbar switch fabric," in *Proc. IEEE ICC*, Helsinki, Finland, Jun. 2001, pp. 1586–1591.

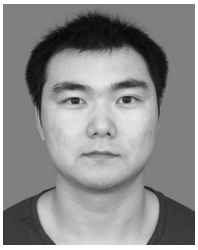
[30] M. Hosaagrahara and H. Sethu, "Max-min fair scheduling in input-queued switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 4, pp. 462–475, Apr. 2008.

[31] D. A. Heger, "A disquisition on the performance behavior of binary search tree data structures," *Eur. J. Inf. Professional*, vol. 5, no. 5, pp. 67–75, Oct. 2004.

Bing Hu (S'06–M'10) received the B.Eng. and M.Phil. degrees in communication engineering from the University of Electronic Science and Technology of China in 2002 and 2005, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, The University of Hong Kong, in 2009. He joined the College of Information Science and Electronic Engineering, Zhejiang University in 2010, where he is currently an Associate Professor. His current research interests include next-generation Internet, high-speed packet switch/router design, and datacenter networks.

Kwan L. Yeung (SM'99) was born in 1969. He received the B.Eng. and Ph.D. degrees in information engineering from The Chinese University of Hong Kong in 1992 and 1995, respectively. He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong, in 2000, where he is currently a Professor. His research interests include next-generation Internet, packet switch/router design, all-optical networks, datacenter networks, and wireless data networks.

Qian Zhou received the bachelor's degree in information and communication engineering from Zhejiang University, China, in 2013, where she is currently pursuing the master's degree in electronic and communication engineering. Her research interests mainly focus on routing and scheduling algorithms for high-speed switches.



Chunzhi He received the B.Eng. and master's degrees in communication engineering from the University of Electronic Science and Technology of China in 2006 and 2009, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, The University of Hong Kong, in 2014. His research interests include next-generation Internet, high speed packet switch/router design, and data center networks.