Design and Performance Analysis of a Practical Load-Balanced Switch

Yanming Shen, Shivendra S. Panwar, and H. Jonathan Chao

***Abstract*—The load-balanced (LB) switch proposed by C.S. Chang *et al*. [1], [2] consists of two stages. First, a load-balancing stage converts arriving packets into uniform traffic. Then, a forwarding stage transfers packets from the linecards to their final output destination. Load-balanced switches do not need a centralized scheduler and can achieve 100% throughput for a broad class of traffic distributions. However, load-balanced switches may cause packets at the output port to be out of sequence. Several schemes have been proposed to tackle the out-of-sequence problem of the load-balanced switch. They are either too complex to implement, or introduce a large additional delay. In this paper, we present a practical load-balanced switch, called the Byte-Focal switch, which uses packet-by-packet scheduling to significantly improve the delay performance over switches of comparable complexity. We prove that the queues at the input need only finite buffering, and that the overall switch is stable under any traffic matrix. Our analysis shows that the average queuing delay is roughly linear with the switch size N, and although the worst case resequencing delay is N^2, the average resequencing delay is much smaller. This means that we can reduce the required resequencing buffer size significantly.**

***Index Terms*—Load-balanced switches, throughput.**

I. Introduction

INTERNET traffic continues to grow rapidly. To keep pace with the demand, there has been a significant research effort on high-speed large-capacity packet switch architectures that out-perform today's switch architectures. Because of the memory speed constraint, most proposed large-capacity packet switches use input buffering alone or in combination with other schemes, such as output buffering or cross-point buffering. Input-buffered switches are required to resolve output contention, that is, find a match between inputs and outputs per time slot (see e.g., [3]–[7]). Thus, the issue of how to schedule packets efficiently to achieve high throughput and low delay for a large-capacity switch has been one of the main research topics in the past few years. Although several practical scheduling schemes have been proposed or implemented (for instance [8]–[11]), most of them require a centralized packet scheduler, increasing the interconnection complexity between the line cards and the packet scheduler, while some schemes need a speedup of up to two [12],[13] to compensate for deficiencies in packet scheduling. Due to the difficulty of scheduling its switch fabric, commercial high-speed switches typically cannot guarantee 100% throughput for all arrival traffic patterns, and this goal will become even more difficult in the future as the number of interfaces and the interface line speeds increase.

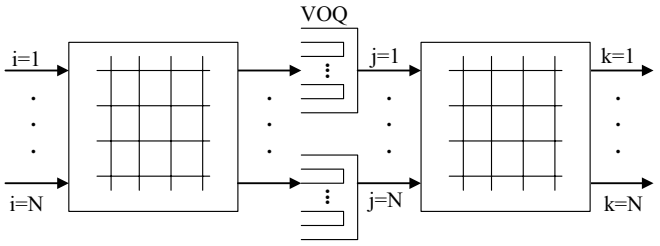

Fig. 1. The architecture of the load-balanced Birkhoff-von Neumann switch.

The load-balanced Birkhoff-von Neumann switch architecture (LB-BvN) [1], [2] introduced by C.S. Chang *et al*. addressed the two main problems commonly present in previous switch architectures: centralized scheduling and the lack of throughput guarantees. This makes the load-balanced switch an appealing architecture to study. This architecture is based on spreading packets uniformly ("load-balancing") inside the switch before forwarding them to their correct destination. The LB-BvN switch does not require a centralized scheduler and is thus highly scalable. At the same time, it can guarantee 100% throughput for a broad class of traffic.

As shown in Fig. 1, the basic LB-BvN switch consists of two identical switches and does not need any scheduler. Each of the two switching stages goes through the same predetermined cyclic shift configuration. Therefore, each input is connected to each output exactly $\frac{1}{N}$th of the time, regardless of the arriving traffic. The first stage is a load-balancer that spreads traffic over all the second stage Virtual Output Queues (VOQs). The second stage is an input-queued crossbar switch in which each VOQ is served at a fixed rate. The first stage supplies the second stage with a uniform distribution by performing load-balancing using a deterministic connection pattern. Since the second stage switch receives uniform traffic, it can achieve 100% throughput with a fixed periodic connection.

In this paper, we propose a practical load-balanced switch architecture, the *Byte-Focal* switch, which uses a resequencing buffer to solve the out-of-sequence problem. We call the

Paper approved by T.-S. P. Yum, the Editor for Packet Access and Switching of the IEEE Communications Society. Manuscript received September 25, 2007; revised April 15, 2008 and July 22, 2008.

Y. Shen is with the Department of Computer Science and Engineering, Dalian University of Technology, Dalian, Liaoning, 116023 China (e-mail: syanming@photon.poly.edu).

S. S. Panwar and H. J. Chao are are with the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, Brooklyn, NY, 11201 USA (e-mail: panwar@catt.poly.edu, chao@poly.edu).

This work is supported in part by the National Science Foundation under Grant CNS-0435303, and also in part by the New York State Center for Advanced Technology in Telecommunications (CATT).

Earlier versions of the results in this paper appeared in the Proceedings of the IEEE High Performance Switching and Routing Workshop 2005 & 2006.

Digital Object Identifier 10.1109/TCOMM.2009.08.070477

switch *Byte-Focal* to reflect the fact that packets belonging to a flow (traffic from an input to an output) are spread to all line cards and brought to a focal point (the destined output). The Byte-Focal switch is simple to implement and highly scalable. It does not need a complex scheduling algorithm, or indeed any communication between linecards, while achieving 100% throughput.

The rest of the paper is organized as follows. Section II discusses other load-balanced switch designs. Section III presents the Byte-Focal switch architecture and various scheduling schemes at the first stage. In Section IV, we prove 100% throughput and analyze the average delay in the Byte-Focal switch. In Section V, we run extensive simulations to study the delay performance of the Byte-Focal switch. Section VI concludes the paper.

II. Related Work

Though load-balanced switches can achieve 100% throughput without centralized scheduling, the FIFO discipline, which needs to be maintained for every traffic flow between input-output pairs, might be violated in the load-balanced switch. In its simplest form, the switch mis-sequences packets by an arbitrary amount. Several solutions have been proposed to tackle the unbounded out-of-sequence problem and can be categorized into two different approaches. The first approach is to prevent packets from being received out-of-sequence at the outputs (e.g., FFF (Full Frames First) [14], Mailbox switch[15]). The second approach is to limit the number of slots a packet can be out of sequence to an upper bound, e.g., $O(N^2)$, and then add a resequencing buffer (RB) at the output to reorder the packets. Such schemes include FCFS (First Come First Serve) [16], EDF (Earliest Deadline First) [16], and FOFF (Full Ordered Frames First) [17].

The FFF scheme proposed in [14] is a frame-based (where a frame slot equals N time slots) scheduling algorithm, which uses a 3 Dimension Queue (3DQ) between the two switch fabrics to resolve the out-of-sequence problem. The 3DQ distinguishes packets among all flows from input i to output k via a center buffer j. As a result, packets belonging to different flows do not interleave at the same central buffer queue and packets arrive at output port k in-order without any need for resequencing. In order to efficiently deliver packets from inputs to outputs, the FFF uses the Full Frames First scheduling policy to maintain system performance. However, such a scheduling algorithm requires buffer communications across stages. The large communication overhead may become the bottleneck for a large scale switch using FFF.

The Mailbox switch in [15] uses virtual waiting time to coordinate the packet departures. It prevents the out-of-sequence problem by simply forcing any packet departure from the central buffer (mailbox) not to be earlier than its predecessor. Although the Mailbox switch prevents packets from being out-of-sequence without using a resequencing buffer, it suffers from low buffer utilization at the center stage. With a limited amount of buffering, the Mailbox switch can only achieve 75% throughput. The authors of [15] also suggested increasing the Mailbox switch's throughput by allowing a limited amount of out-of-sequence packets, and adding a corresponding amount of resequencing buffer at the outputs. This modification boosts the Mailbox switch's throughput from 75% to 95% at the cost of additional resequencing buffers. In [18], a simple load-balanced switch architecture was proposed, where the resequencing buffer size needs to be proportional to the central buffer size to avoid the out-of-sequence problem. The reduced buffer size requirement is at the cost of losing some throughput as shown in the simulations.

In [16], a load-balanced switch with multi-stage buffering is proposed to solve the out-of-sequence problem. There, load-balancing buffers are added before the first switch, and resequencing and output buffers are inserted after the second switch. Two scheduling policies in the central stage are presented in [16]: the First Come First Serve (FCFS) policy and the Earliest Deadline First (EDF) policy. For the FCFS scheme, a jitter control mechanism is introduced in the VOQ in front of the second stage. Such a jitter control mechanism delays every packet to its maximum delay at the first stage so that every packet has the same delay before entering an input port of the second stage. However, this scheme increases the average delay. In the EDF scheme, every packet is assigned a deadline that is the departure time from the corresponding FCFS output-buffered switch. Packets are scheduled according to their deadlines in the central buffers. Although the EDF scheme can greatly reduce the average delay, it needs to retrieve the packet with the smallest time stamp from a queue, making it hard to implement in a high speed switch.

The Full Ordered Frames First (FOFF) scheme is presented in [17]. The FOFF scheme bounds the difference in lengths of the VOQs in the second stage, and then uses a resequencing buffer at the third stage. It is very challenging to design scheduling algorithms in high-speed switches given the short time interval (e.g., if the link speed is 160 Gbps and the cell length is 64 bytes, then each time slot is about 3 ns). The FOFF scheme performs frame-based scheduling to relax this timing constraint, but at the cost of wasting bandwidth. We believe that with the low complexity scheduling algorithms presented in this paper, packet-by-packet scheduling used by Byte-Focal is still feasible in high-speed switches.

III. The Byte-Focal Switch Architecture

The Byte-Focal switch is based on packet-by-packet scheduling to maximize the bandwidth utilization of the first stage and thus improve the average delay performance. Fig. 2 shows the Byte-Focal switch architecture. It consists of two deterministic switch fabrics and three stages of queues, namely input queue i, center stage queue j, and output resequencing buffer (RB) k, where $i, j, k = 1, 2, \ldots, N$. Both switch fabrics use a deterministic and periodic connection pattern. Thus, at the first stage, at any time slot t, the connection pattern (i, j) is given by

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

$i = 1, \ldots, N$ and $j = 1, \ldots, N$.

Similarly, the connection pattern (j, k) at the second stage satisfies

$$j = (k + t) \bmod N, \tag{2}$$

$j = 1, \ldots, N$ and $k = 1, \ldots, N$.

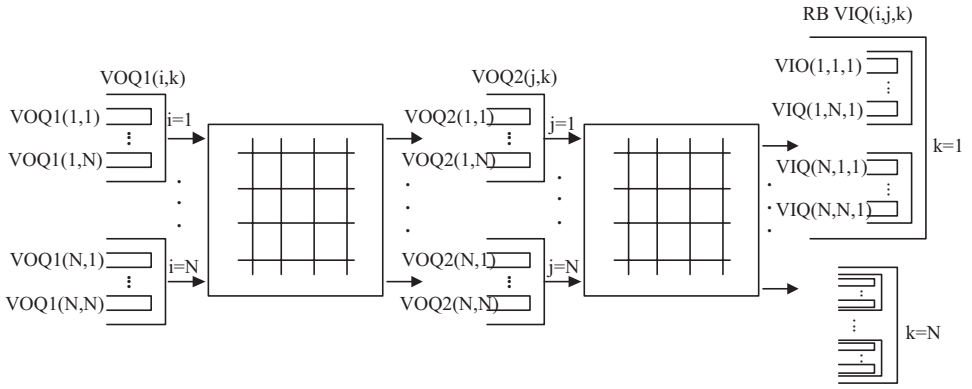

Fig. 2. The Byte-Focal switch architecture.

There are two stages of VOQs in the Byte-Focal switch, VOQ1 and VOQ2 for the first and second stage switch, respectively. We define the flow f_{ik} as the packets arriving at the input port i and destined to output port k. Packets from f_{ik} are placed in VOQ1(i,k). Since, at each time slot, the input port at the first stage is connected to the second stage cyclically, the packets in VOQ1(i,k) are sent to the N second stage input ports in a round-robin manner (as explained next) and are placed in VOQ2$(1,k)$, VOQ2$(2,k)$, ..., VOQ2(N,k) according to their final destination.

Consider VOQ1(1,1) as an example. When input 1 at the first stage is connected to input 1 at the second stage, assume VOQ1(1,1) is selected for service by the first stage scheduling algorithm (explained in Section III-A), then the head-of-line packet in VOQ1(1,1) is transfered to VOQ2(1,1). The Byte-Focal switch requires that packets in a flow are sent to the second stage cyclically, i.e., the packets are sent in the order of the second stage input 1, input 2, ..., input N. Therefore, the next packet in VOQ1(1,1) should be sent to input 2 at the second stage, that is, VOQ2(2,1). From (1), at the next time slot, input 1 at the first stage is connected to input 2 at the second stage, then VOQ1(1,1) can transfer a cell to VOQ2(2,1) if it is selected for service. If it is not selected for service in this time slot, then it can only get its next opportunity to be served when input 1 at the first stage is connected to input 2 at the second stage the next time, i.e., after N time slots. Therefore, we can see that the Byte-Focal switch guarantees that the cumulative number of packets sent to each second stage input port for a given flow differs by at most one. The VOQ2 will then be served by the second fixed, equal-rate switch. Since the packets, in general, suffer different delays in the second stage, they arrive at the output out of order.

A. First Stage Scheduling Algorithm Design

In improving the average delay performance, the scheduling scheme at the first stage plays a very important role in the Byte-Focal switch. Since the packets in VOQ1(i,k) are cyclically distributed to the second stage, as a result, when the first stage input port i is connected to the second stage input port j, only some of the VOQ1s can be served. Among all the candidate VOQ1s, we need to pick a VOQ1 at i to send packets to j. To this end, each VOQ1(i,k) has a *J pointer* that keeps track of the last second stage input to which a packet was transferred. When the first stage input i is connected to the second stage input j, a VOQ1(i,k) can be served only if its *J pointer* is pointing to the second stage input j. If VOQ1(i,k) is selected for service, then its *J pointer* will point to the next second stage input, i.e., $(j+1) \text{mod} N$. Then the first stage scheduling problem can be stated as follows:

> When input i is connected with j, each VOQ1(i,k) whose J pointer value is equal to j sends a request to the arbiter, and the arbiter selects one of them to serve.

As we shall see, the Byte-Focal switch performs the above first stage scheduling independently at each input port using locally available information. Thus, it does not need any communication between different linecards.

Let $P_{ik}(t)$ be the J pointer value for VOQ1(i,k) at time t. Define a set $S_j(t) = \{\text{VOQ1}(i,k) | P_{ik}(t) = j\}$. $S_j(t)$ is thus the set of VOQ1s that can send packets to the second stage input j at time t. To achieve small delay while maintaining fairness among all traffic flows, an efficient arbitration is necessary to schedule the departure of the head-of-line (HOL) packets of the VOQ1s. We will next consider four ways of picking a VOQ1 to serve from the set $S_j(t)$.

1) Round-Robin: One simple way to do the first stage scheduling is to use the *round-robin* scheme. To this end, the arbiter at each of the first stage inputs maintains a *round-robin pointer*. When VOQ1(i,k) is served in a time slot, the round-robin pointer will move to the next VOQ1, i.e., $(k+1) \text{mod} N$. With the round-robin scheme, the arbiter at each input port selects one from the set $S_j(t)$ in round-robin order starting from the current pointer position. The round-robin mechanism has a complexity of $O(1)$ and is easy to implement. However, it turns out that the round-robin scheme cannot guarantee 100 % throughput. Next, we will present three algorithms which provably guarantee 100% throughput.

2) Longest Queue First: Although the round-robin arbitration achieves fairness among all the traffic flows, under non-uniform traffic conditions, some congested VOQ1s could overflow and the system becomes unstable (see the simulation results in Fig. 5). In order to stabilize the system, we need to give high priority to the congested VOQ1s. The *longest queue first* algorithm ensures that, at each time slot, the arbiter at each input port chooses to serve the VOQ1 from the set $S_j(t)$ with the longest queue. The longest queue first scheme can

achieve good performance, and finding the longest queue has a complexity of $O(\log N)$.

3) Fixed Threshold Scheme: To reduce the complexity of the longest queue first scheme, instead of picking the longest queue, it is easier to identify the congested VOQ1s by observing if their queue length exceeds a predetermined threshold (TH), which we pick to be equal to the switch size N. Let $q_{ik}(t)$ be the length of the VOQ1(i,k), and $q_{il}(t)$ be the length of the VOQ1(i,l) served in the previous time slot. Define a subset of $S_j(t)$ as follows:

$$S_j^{'}(t) = \{\text{VOQ1}(i,k) | \text{VOQ1}(i,k) \in S_j(t) \text{ and } q_{ik}(t) \geq TH\},$$

then $S_j^{'}(t)$ is the set of VOQ1s that have more than TH cells and can send cells to j. The *fixed threshold* algorithm is:

1) At each time slot, if $q_{il}(t) \geq TH$, continue to serve this queue.
2) If not, the arbiter picks round-robin among the queues in set $S_j^{'}(t)$ starting from VOQ1(i,l).
3) If $S_j^{'}(t)$ is empty and $q_{il}(t) > 0$, then it keeps serving the queue corresponding to $q_{il}(t)$.
4) If $q_{il}(t) = 0$, pick round-robin among the queues in set $S_j(t)$, starting from VOQ1(i,l).

As compared to the longest queue first scheme, the fixed threshold scheme only needs to check if the queue length is greater than the threshold which has a complexity of $O(1)$.

4) Dynamic Threshold Scheme: For the fixed threshold scheme, we prove that it can achieve 100% throughput if we set the threshold to be the switch size N. However, with threshold N, as the switch size becomes large, it will cause large average delays. The reason is that the VOQ1 length has to reach N before being identified as congested. Before reaching the threshold, it competes with other VOQ1s that have much smaller queue lengths. To better identify congested queues under different switch sizes and different traffic loadings, we propose the *dynamic threshold* scheme. We set the dynamic threshold value (TH) to $Q(t)/N$, where $Q(t)$ is the total VOQ1 queue length at an input port at time t. $Q(t)/N$ is therefore the average VOQ1 queue length. The dynamic threshold scheme operates in the same way as the fixed threshold scheme except that the threshold is now set to the average queue length for that input. To this end, each input port maintains a counter to keep track of the total number of packets at an input port. The dynamic threshold scheme has the same complexity as the fixed threshold scheme, i.e., $O(1)$ since a counter does not change the complexity order.

Note that for the fixed and dynamic threshold schemes, one can also set different thresholds. However, by setting the thresholds as described above, we can prove, in the next section, that the two schemes can achieve 100% throughput, which is not generally true for other threshold values.

B. Resequencing Buffer Design

To solve the out-of-sequence problem, the Byte-Focal switch uses a resequencing buffer (RB), where the virtual input queue (VIQ) structure is applied (as shown in Fig. 2).

At each output, there are N sets of VIQs, where each set corresponds to an input port i. Within each VIQ set, there are N queues with each queue corresponding to a second stage input j. VIQ(i,j,k) separates each flow not only by its input and output ports i and k, but also by its second stage queue j. Packets from input i destined to output k via second stage input j are stored in VIQ(i,j,k), and it is easy to see that the packets in the same VIQ(i,j,k) are in order.

We define the head of flow (HOF) packet as the first packet of a given flow that has not yet left the switch, and the head of line (HOL) packet as the first packet of a given VIQ(i,j,k) queue. In each VIQ set, a pointer points to the VIQ(i,j,k) at which the next expected HOF packet will arrive. Because of the service discipline of the first stage switch, each input port evenly distributes packets in round-robin order into the second stage queues. This guarantees that the head-of-flow (HOF) packet will appear as a head-of-line (HOL) packet of a VIQ set in round-robin order. Therefore, at each time slot, if the HOF packet is in the output, it is served and the pointer moves to the next HOF packet location VIQ$(i,(j+1) \bmod N,k)$.

Since there are potentially up to N HOF packets available at any time slot, each corresponding to one of N input ports, more than one packet may be eligible to be served in the same time slot. Therefore, in addition to the VIQ structure, there is a departure queue (DQ) with a length of at most N entries that facilitates round-robin service discipline. The DQ is simply a FIFO queue. It stores the indices of the VIQ$(i,\cdot,k)$ sets, but at most one from each VIQ$(i,\cdot,k)$ set. When the HOF packet of VIQ$(i,\cdot,k)$ set i arrives, index i joins the tail of the DQ. When a packet departs from the DQ, its index is removed from the head of the DQ and joins the tail of the DQ if its next HOF packet has arrived. The advantage of using the VIQ and the DQ structure is that the time complexity of finding and serving packets in sequence is $O(1)$. At each time slot, each VIQ$(i,\cdot,k)$ set uses its pointer to check if the HOF packet has arrived, while the output port serves one packet from the head of the DQ. As explained above, the VIQ structure ensures that the Byte-Focal switch will emit packets in order.

IV. Throughput and Delay Analysis

A. 100% Throughput

In this section, we will show that the Byte-Focal switch with the longest queue first, fixed threshold and dynamic threshold schemes can achieve 100% throughput. The round-robin scheme does not have this property.

Definition 1: If Q is the total queue length of a system, then the system is said to be stable if $E\{Q\} < \infty$.

First, we can see that the queue length at the second stage is bounded. Since the traffic arrivals to the second stage are uniform, if the arrival rate to an output port is λ, then the arrival rate to an individual VOQ is λ/N. The second stage uses a deterministic cyclic configuration, which means that the service rate for a VOQ is $1/N > \lambda/N$. Therefore, the second stage queue is bounded. For the resequencing buffer, it has been shown in [17] that the maximum resequencing buffer size is N^2. Therefore, we only need to show that the queue length at the first stage is also bounded.

Let $Q_i(t)$ and $q_{ik}(t)$ represent the total queue length at an input port i and the individual VOQ1(i,k) length at time t, respectively. Let $q_{is}(t)$ denote the length of VOQ1(i,s) being

served at time t. Note that VOQ1(i,s) can mean different VOQ1s, i.e., at time t, it may correspond to VOQ1(i,k_1) that is being served, and in the next slot, it may correspond to a different VOQ1(i,k_2) that is then being served, depending on the specific scheduling algorithm.

Lemma 1: If $q_{is}(t_0) < N$ and $Q_i(t) \geq N(N-1)+1$ for $t \geq t_0$, then it takes at most $N-1$ time slots to find a VOQ1(i,k) with $q_{ik}(t) \geq N$ to serve.

The proof of this Lemma is presented in the Appendix.

Next we define a term, $Q_i(t)$ work-conserving, that we will use.

Definition 2: $Q_i(t)$ work-conserving: an input port i is $Q_i(t)$ work-conserving if it can be idle only if there are less than $Q_i(t)$ packets at that input port.

With this definition, we have the following Lemma, the proof of which can be found in the Appendix.

Lemma 2: If at time t_0, $q_{is}(t_0) \geq N$, then input i at the first stage is $N(N-1)+1$ work-conserving.

From Lemma 1 and Lemma 2, we can prove that the input queue length is bounded as stated in the following theorem.

Theorem 1: The Byte-Focal switch with the longest queue first, fixed and dynamic threshold schemes are stable for any input traffic, and the input queue length cannot exceed N^2.

Proof: If initially all VOQ1s are empty, when $Q_i(t) = N(N-1)+1$ for the first time, from Lemma 1, then it wastes at most $N-1$ time slots to find a queue length greater than N to serve. After that, from Lemma 2, the system is $N(N-1)+1$ work-conserving. Since there is at most one arrival in a time slot, therefore, $Q_i(t)$ is upper bounded by

$$N(N-1)+1+N-1 = N^2.$$

■

We have shown that the queue length at the first stage is upper bounded by N^2. The queue lengths at the second stage and the resequencing buffer are also bounded as explained earlier in this Section. Therefore, we conclude that the Byte-Focal switch is stable.

B. Average Delay Analysis

A packet in the Byte-Focal switch experiences queuing delays at the first and second stage, and also the resequencing delay at the output. In this section, we analyze the average delay of the Byte-Focal switch under uniform traffic. Throughout the analysis, we assume a uniform Bernoulli i.i.d traffic model with a rate of λ.

1) Second Stage Delay: Consider a VOQ2(j,k) and $a_{jk}(t)$ is the arrival to VOQ2(j,k) in time slot t. The input j at the second stage is connected to output k once in every N time slots. The arrivals to this queue can be approximated by a Bernoulli process with a rate of $\frac{\lambda}{N}$. Assume that at time T, the VOQ2(j,k) is connected to output k. Then we have the following recursive equation for VOQ2(j,k):

$$q_{jk}(T+n) = q_{jk}(T) + \sum_{s=1}^{n} a_{jk}(T+s),$$
$$n = 1, 2, \ldots, N-1.$$
$$q_{jk}(T+N) = \max\{q_{jk}(T) + \sum_{s=1}^{N} a_{jk}(T+s) - 1, 0\}. \quad (3)$$

By solving this recursion as shown in the Appendix, we have

$$E\{q_{jk}(t)\} = \frac{N-1}{N}\frac{\lambda}{2(1-\lambda)}. \quad (4)$$

Applying Little's formula, the average queuing delay at the second stage is

$$d_2 = \frac{N-1}{2(1-\lambda)}. \quad (5)$$

However, since the queue length difference is bounded by N [19], when the Byte-Focal switch is heavily loaded, the average delay would be smaller than (5). Consider the system formed by all second stage queues. When the total number of packets destined to output k for all inputs is greater than N^2-N+1, then no queue is empty, and the system is work-conserving and behaves like an output-queued switch. Thus we have

$$q_k(t+1) = \max\{q_k(t) + a_k(t+1) - 1, 0\},$$

where $q_k(t) = \sum_{j=1}^{N} q_{jk}(t)$. The average queuing delay for an output-queued switch is simply

$$\frac{\lambda}{2(1-\lambda)}\frac{N-1}{N}.$$

Table I shows the simulation results and the analysis for the second stage queuing delay with a switch size of $N = 16$. In the simulation, we use the round-robin policy. From the table, we can see when the load is low (less than 0.59), the average delay matches $\frac{N-1}{2(1-\lambda)}$ very well. However, when the loading is extremely high (0.99), the second stage queuing delay behaves like an output-queued switch.

2) First Stage Delay: Consider the first stage under uniform traffic. On the average, each VOQ1 is served once every N time slots, and the queues will behave similarly to a TDM system as in the second stage. Each VOQ1 has Bernoulli arrivals with rate $\frac{\lambda}{N}$ and an approximately deterministic service time of N time slots. Using the same derivation as in the second stage, the queuing delay at the first stage under uniform traffic is

$$d_1 = \frac{N-1}{2(1-\lambda)}. \quad (6)$$

Table I shows the average first stage queuing delay for round-robin scheme for a switch size of $N = 16$. Note that the total queue length at the first stage is bounded by N^2 under longest queue first, fixed and dynamic schemes. Therefore, when the switch is under heavy load, the average delay would be significantly smaller than $\frac{N-1}{2(1-\lambda)}$.

3) Resequencing Delay: To analyze the resequencing delay, consider a tagged packet (TP) [20] in flow f_{ik}. TP arrives at the second stage input j at frame time F_j (one frame time slot = N time slots). Because of the FCFS discipline, the packets which arrive at the switch after the TP cannot affect the departure time of TP from the resequencing buffer. So we only need to consider the packets which were in the switch when TP arrived. The packets in the same flow are cyclically distributed to the N intermediate second stage inputs, from 1 to N. The packets in the same VOQ2 are in order. The resequencing delay(RD) for the TP is only decided by the $N-1$ packets ahead of it, and these $N-1$ packets are sent to the second stage queues $j+1, \ldots, N, 1, \ldots, j-1$ respectively,

TABLE I
FIRST AND SECOND STAGE QUEUING DELAY

	Load	0.09	0.19	0.29	0.39	0.49	0.59	069	0.79	0.89	0.99
Second stage	Simulation	8.23	9.21	10.40	11.87	13.73	16.11	19.22	23.34	29.26	49.16
	$\frac{N-1}{2(1-\lambda)}$	8.24	9.26	10.56	12.3	14.71	18.29	24.19	35.71	68.18	750
	$\frac{\lambda}{2(1-\lambda)}\frac{N-1}{N}$	0.05	0.11	0.19	0.30	0.45	0.67	1.04	1.76	3.79	46.41
First stage	Simulation	8.23	9.33	10.98	13.55	17.75	24.59	35.61	53.03	86.32	322.47

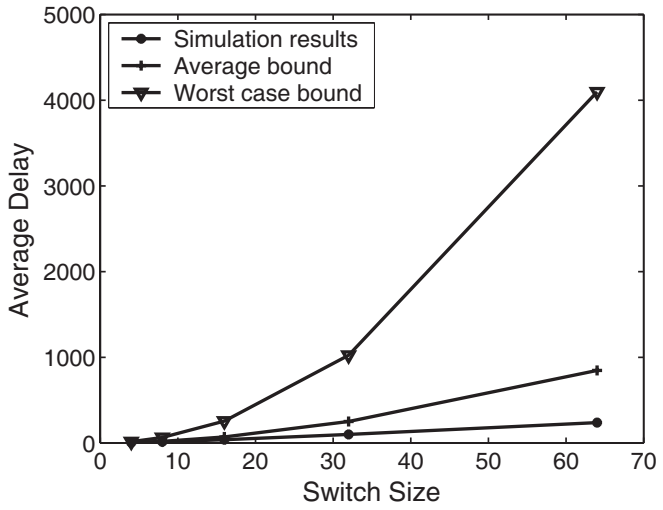

Fig. 3. Resequencing delay for different switch sizes.

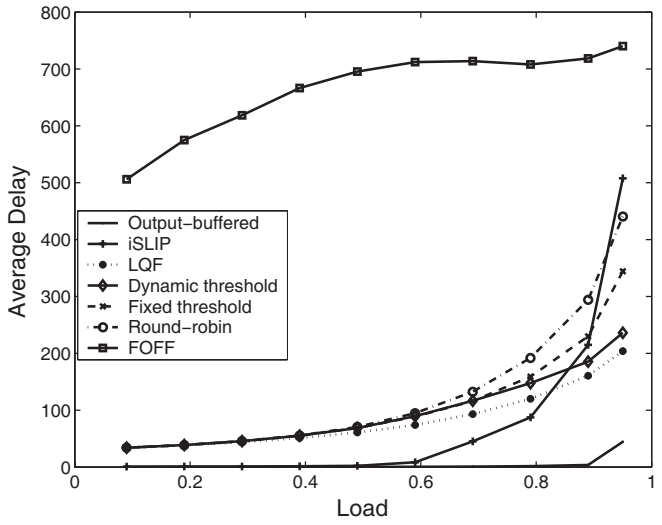

Fig. 4. Average delay under uniform traffic.

which are denoted with $P_{j+1}, \ldots, P_{j-1}$. Let F_l denote the frame time that the packet arrived at second stage queue l, and $\hat{F}_l$ be the time that the packet leaves second stage queue l. Then the resequencing delay for the TP is

$$RD = \max(0, \max_l(\hat{F}_l - \hat{F}_j)), \tag{7}$$

where $l = j+1, \ldots, j-1$.

Then, as shown in the Appendix, the average resequencing delay bound is

$$E\{RD\} \le E\{(\max Y_l)^+\},\ l = 1, 2, \ldots, N-1, \tag{8}$$

where $(X)^+ = \max\{0, X\}$. Y_l can be approximated as a normal distribution with mean $\eta = 0$ and variance $\sigma^2 = 2Np(1-p) = 2\lambda[1 - \lambda/N(1-\lambda/N)]$.

Then from (8), we can get the average resequencing delay bound. Fig. 3 shows how the average resequencing delay grows with N. We can see that the average resequencing delay is roughly linear with the switch size N, and it is much smaller than N^2.

V. SIMULATION STUDIES

Before presenting the switch performance, we outline the simulation settings that will be used to test the various scheduling algorithms. In our simulations, we assume the switch size $N = 32$, unless otherwise noted, and all inputs are equally loaded on a normalized scale $\lambda \in (0, 1)$, and use the following traffic scenarios to test the performance of the Byte-Focal switch:

Uniform i.i.d: $\lambda_{ik} = \lambda/N$.

Diagonal i.i.d: $\lambda_{ii} = \lambda/2$, $\lambda_{ik} = \lambda/2$, for $k = (i+1) \bmod N$. This is a very skewed loading, since input i has packets only for outputs i and $(i+1) \bmod N$.

Hot-spot i.i.d: $\lambda_{ii} = \lambda/2$, $\lambda_{ik} = \lambda/2(N-1)$, for $i \neq k$.

Normally, for single stage switches, the performance of a specific scheduling algorithm becomes worse as the loadings become less balanced.

We compare the average delay induced by different algorithms. As seen in Fig. 4, the frame-based scheduling scheme, FOFF, has a much larger delay. The reason is that FOFF wastes bandwidth whenever a partial frame is sent. At low traffic load, many frames will be sent as partial frames, resulting in considerable bandwidth wastage at the first stage. From the figure, we can see that at low load, the delay difference between FOFF and the Byte-Focal switch is quite large. The Byte-Focal switch performs packet-by-packet scheduling instead of frame-based scheduling, so it reduces the bandwidth wastage. At high traffic load, the Byte-Focal switch also achieves better performance than the FOFF. A single stage switch like iSLIP has a smaller average delay than Byte-Focal when the load is low. But when the switch is heavily loaded, the Byte-Focal switch distributes the traffic evenly to the second stage, thus dramatically reducing the average delay.

Fig. 5 shows the average delay of various schemes under hot-spot loading. Although the round-robin scheme is simple to implement, it is not stable under non-uniform loadings (as seen in the figure, where the throughput is only about 30%). For reference, we have also provided the performance of a typical single stage switch, HE-iSLIP [21] and the ouput-buffered switch. The LQF scheme has the best delay per-

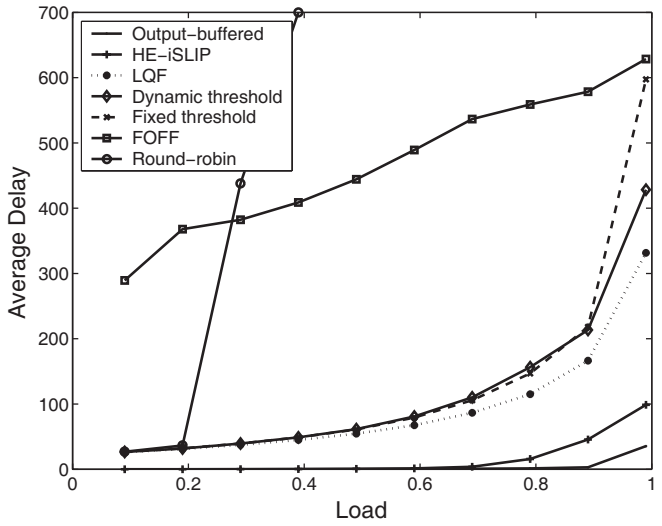

Fig. 5. Average delay under hot-spot loading.

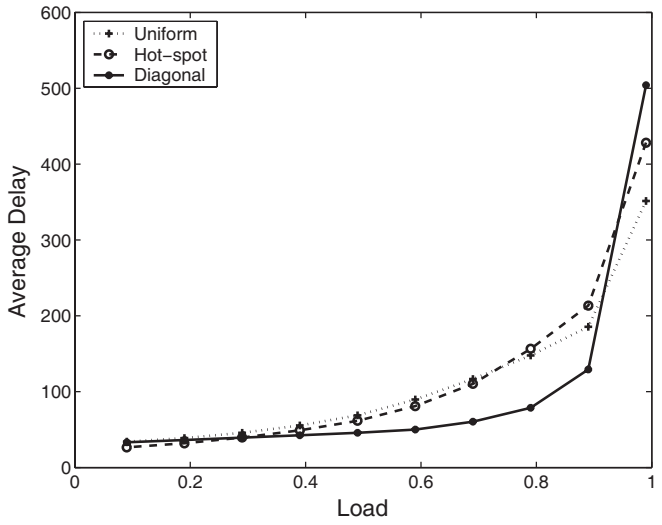

Fig. 6. Average delay for the dynamic threshold scheme.

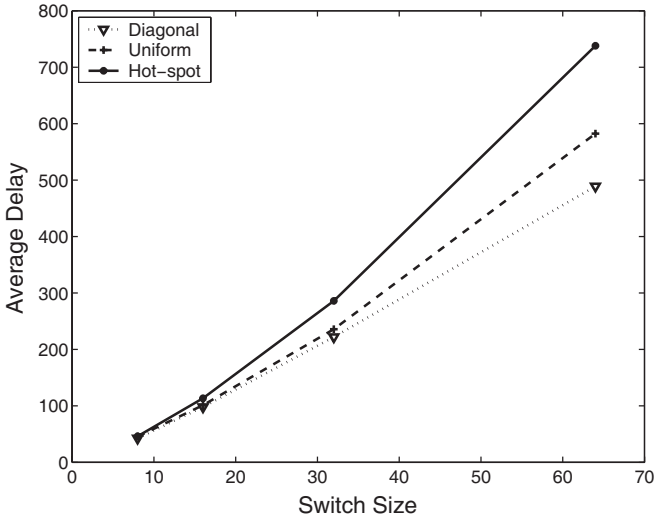

Fig. 7. Average delay vs. switch size N.

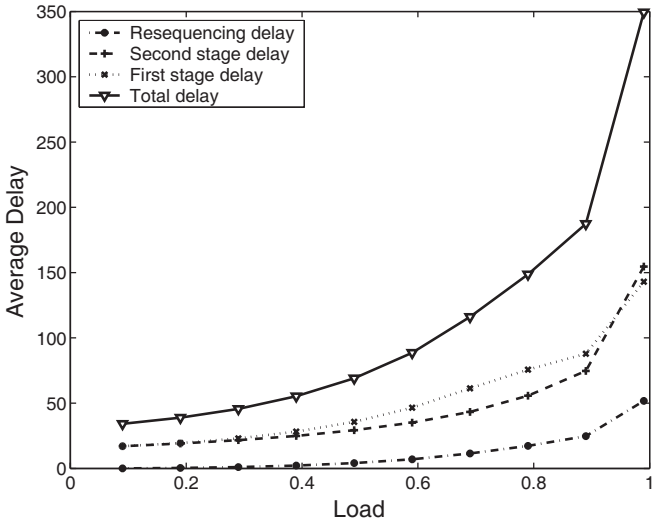

Fig. 8. 3-stage delays under uniform traffic for the dynamic threshold scheme.

formance among the Byte-Focal switch schemes, but, unlike the fixed and dynamic threshold schemes, it is not practical due to its high implementation complexity. From Fig. 5, we can see that the dynamic threshold scheme performance is comparable with the LQF scheme. Compared to the fixed threshold scheme, the dynamic threshold scheme can adapt to the changing input loadings, thus achieving a better delay performance, while maintaining low complexity. We will therefore focus our attention on the dynamic threshold scheme from now on.

In Fig. 6, we study the average delay performance of the dynamic threshold scheme under different input traffic scenarios. As the input traffic changes from uniform to hot-spot to diagonal (hence less balanced), the dynamic threshold scheme can achieve good performance, especially for the diagonal traffic. Diagonal loading is very skewed and difficult to schedule using a centralized scheduling architecture. We also tried the Log-diagonal input matrix[7], and the delay performance is comparable to hot-spot loading. The Byte-Focal switch performs load-balancing at the first stage, thus achieving good performance even under extreme non-uniform loadings. This greatly simplifies traffic engineering design, given the insensitivity of delay to the traffic matrix.

Fig. 7 shows the average delays for the dynamic threshold scheme with different switch sizes with the load kept fixed at 0.95. As shown in the figure, under the input traffic models that we considered, the delay increases as the switch size increases, and the average delays are almost linear with the switch size. Since the Byte-Focal switch does not use a centralized scheduler, it can scale well, unlike switches using centralized schedulers (e.g., iSLIP, HE-iSLIP, etc.), and can achieve good performance even for very large switch sizes.

A cell in the Byte-Focal switch experiences queuing delays at the first stage and second stage, and resequencing delay at the output. Fig. 8 shows the three components of the total delay. As can be seen, the first stage queuing delay and the second stage queuing delay are comparable, and the resequencing delay is much smaller compared to the other two delays.

Since Internet traffic is bursty [22], we also study the delay

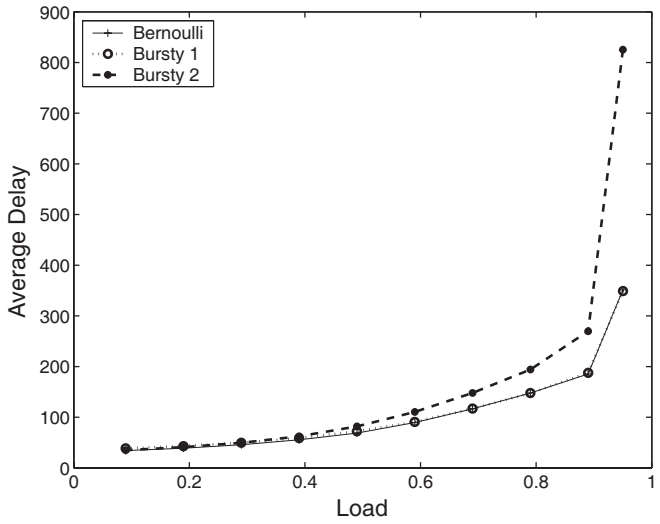

Fig. 9. Average delay of the dynamic threshold scheme under bursty traffic.

performance under bursty traffic. Consider the same simulation settings, but now the packets arrive in bursts. The average burst length is set to be 10 cells. At a particular input port, after a burst, the probability that there is another arriving burst is μ, and the probability that there is no packet arriving corresponding to the next burst is $1-\mu$ (then the loading to this input port is $\lambda = \frac{10\mu}{1+9\mu}$). We consider two scenarios:

- Bursty 1: cells within the same burst are uniformly distributed to the N output ports.
- Bursty 2: cells within the same burst go to the same destination, which is uniformly distributed over N output ports.

Fig. 9 shows the average delay of the Byte-Focal switch with the dynamic threshold scheme under the Bernoulli and bursty traffic models. We can see that the average delays under the Bernoulli and Bursty 1 traffic scenario are identical. In comparison with the single stage switches, the Byte-Focal switch achieves considerable burst reduction, therefore it is very effective in reducing the average delay. From our simulations, the delay performance is worse for Bursty 2 as compared to Bursty 1, especially when the traffic load is high.

VI. Conclusion

In this paper, we present a practical high performance load-balanced switch architecture: the Byte-Focal switch. Compared with traditional centralized-scheduler architectures, the load-balanced switch can achieve 100% throughput and does not need a centralized scheduler. Also, it uses only N patterns for the switch fabric out of the possible $N!$ patterns and the pattern sequence is predetermined; this simplifies the switch fabric. In addition to these general properties of load-balanced switches, the Byte-Focal switch has several appealing properties:

1) Every packet leaves the switch in order.
2) It does not need any communication between stages or linecards. This simplifies the switch control and avoids the loss of bandwidth due to the exchange of information.
3) Although a scheduling decision is preformed every time slot, the scheduling algorithm uses locally available information, and is easy to compute.
4) It can achieve a uniformly good delay performance over a wide range of traffic matrices.

The Byte-Focal switch combines low complexity with good performance, allowing it to be scaled up to large N at high line speeds.

Appendix

Proof of Lemma 1: Since $Q_i(t_0) \geq N(N-1)+1$, and $q_{is}(t_0) < N$, therefore there exists a VOQ1(i,k) different from the VOQ1(i,s) with queue length $q_{ik}(t_0) \geq N$. Assume the pointer at VOQ1(i,k) points to the second stage input j, and after T time slots ($T \leq N-1$), input i is connected to j. At time t_0+T, we have

$$q_{ik}(t_0+T) \geq q_{ik}(t_0) \geq N. \tag{9}$$

This means that at time t_0+T, in set $S_j(t_0+T)$, there is a queue with queue length greater than N.

For LQF, the longest queue is chosen in set $S_j(t_0+T)$, therefore, $q_{is}(t_0+T) \geq N$. For the fixed threshold scheme, $S_j^{'}(t_0+T)$ is non-empty, therefore, $q_{is}(t_0+T) \geq TH = N$. For the dynamic threshold scheme, at time t_0+T, there exists a VOQ1(i,k) with queue length $q_{ik}(t_0+T) \geq TH = \frac{Q_i(t_0+T)}{N} \geq N$. With the same reasoning, at time t_0+T, $q_{ik}(t_0+T) \geq TH$, and $S_j^{'}(t_0+T)$ is nonempty, therefore, $q_{is}(t_0+T) \geq N$.

Thus for LQF, fixed threshold and dynamic threshold schemes, we always have $q_{is}(t_0+T) \geq N$, with $T \leq N-1$.

Proof of Lemma 2: Since $q_{is}(t_0) \geq N$, after some time $t_b > t_0$, $q_{is}(t_b)$ might drop below N. We define a cycle from t_b, when $q_{is}(t_b) = N-1$ to $t_e = t_b+T$, when $q_{is}(t_b+T) = N$, which is the duration that the queue being served stays below N. Since $Q_i(t) \geq N(N-1)+1$, for $t \in [t_b, t_e]$, from Lemma 1, we have $T \leq N-1$. Within each cycle, we will show there is no time slot wasted. At the beginning of each cycle, $q_{is}(t_b) = N-1$, and at the end of each cycle, $q_{is}(t_e) = N$.

From the definition of the longest queue first, fixed threshold and dynamic threshold schemes, we can see that the scheduler will only switch to a VOQ1 with longer queue length. Also, since there is at most one departure in a time slot, for any $t_b \leq t < t_e$,

$$q_{is}(t) \geq N-1-(t-t_b).$$

Since $t < t_e$, we have

$$q_{is}(t) > N-1-(t_e-t_b).$$

But $t_e - t_b \leq N-1$, thus there is at least one non-empty queue available for service with $q_{is}(t) > 0$, and there is no time slot wasted. We therefore conclude that the first stage is $N(N-1)+1$ work-conserving.

Derivation of the second stage delay: The recursion (3) has the solution [23]

$$E\{q_{jk}(T)\} = \frac{N-1}{N}\frac{\lambda^2}{2(1-\lambda)}. \tag{10}$$

Then the average queue length is

$$\begin{aligned} E\{q_{jk}(t)\} &= \frac{1}{N}\sum_{n=0}^{N-1} E\{q_{jk}(T+n)\} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} E\{q_{jk}(T)+\sum_{s=1}^{n} a_{jk}(T+s)\} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} [E\{q_{jk}(T)\}+\frac{\lambda}{N}n] \\ &= \frac{N-1}{N}\frac{\lambda^2}{2(1-\lambda)}+\frac{\lambda(N-1)}{2N} \\ &= \frac{N-1}{N}\frac{\lambda}{2(1-\lambda)}. \end{aligned} \tag{11}$$

Derivation of the resequencing delay: Since each VOQ at the second stage is work-conserving at each frame time slot, any packet in f_{ik} that arrives at the second stage input queue j, with queue length $q_{jk}(F_j)$, at time F_j will leave there at

$$\hat{F}_j = F_j + q_{jk}(F_j). \tag{12}$$

Then the resequencing delay becomes

$$RD = \max(0, \max_l[(F_l + q_{lk}(F_l)) - (F_j + q_{jk}(F_j))]). \tag{13}$$

Also, we have

$$\begin{aligned} &F_l + q_{lk}(F_l) - [F_j + q_{jk}(F_j)] = q_{lk}(F_l) - q_{jk}(F_l) \\ &+ F_l - F_j + [q_{jk}(F_l) - q_{jk}(F_j)]. \end{aligned}$$

Since there is at most one departure per frame slot, $F_l - F_j + [q_{jk}(F_l) - q_{jk}(F_j)] \le 0$. Therefore,

$$RD \le \max(0, \max_l q_{lk}(F_l) - q_{jk}(F_l)).$$

At each frame slot, we have, using Lemma 1.3.1 in [24],

$$q_{lk}(F) = \max_{0\le S\le F}\sum_{i=1}^{N}[A_{ilk}(F) - A_{ilk}(S)] - (F-S),$$

where $A_{ilk}(t)$ is the cumulative number of packets flowing from input i to output k via the second stage l. Assume the maximum is achieved at S', then

$$q_{lk}(F) = \sum_{i=1}^{N}[A_{ilk}(F) - A_{ilk}(S')] - (F-S').$$

Similarly,

$$\begin{aligned} q_{jk}(F) &= \max_{0\le S\le F}\sum_{i=1}^{N}[A_{ijk}(F) - A_{ijk}(S)] - (F-S) \\ &\ge \sum_{i=1}^{N}[A_{ijk}(F) - A_{ijk}(S')] - (F-S'). \end{aligned}$$

Therefore,

$$\begin{aligned} q_{lk}(F) - q_{jk}(F) &\le (\sum_{i=1}^{N}[A_{ilk}(F) - A_{ilk}(S')] - (F-S')) \\ &- (\sum_{i=1}^{N}[A_{ijk}(F) - A_{ijk}(S')] - (F-S')) \\ &= \sum_{i=1}^{N}[A_{ilk}(F) - A_{ijk}(F)] - [A_{ilk}(S') - A_{ijk}(S')]. \end{aligned}$$

Since the packets in f_{ik} are sent to the second stage in round-robin order, from 1 to N, without loss of generality, assume l is a lower index than j, then we have

$$A_{ilk}(t) - A_{ijk}(t) = 1 \text{ or } 0.$$

Note that $A_{ilk}(t) - A_{ijk}(t) = 1$ means that from flow f_{ik} there is one more arrival to the second stage l than to the second stage k. Define the steady state values of $P(A_{ilk}(t) - A_{ijk}(t) = 1) = p$ and $P(A_{ilk}(t) - A_{ijk}(t) = 0) = 1 - p$. Since the arrival rate of the flow f_{ik} is λ/N, we approximate p by $\lambda/N(1-\lambda/N)$, which means that there is an arrival to second stage l, but no arrival to second stage k. Define

$$X_i = [A_{ilk}(t) - A_{ijk}(t)] - [A_{ilk}(s) - A_{ijk}(s)], t \ne s,$$

then we have

$$X_i = \begin{cases} -1 & \text{with probability of } p(1-p), \\ 0 & \text{with probability of } p^2 + (1-p)^2, \\ 1 & \text{with probability of } p(1-p). \end{cases}$$

Let $Y_l = \sum_{i=1}^{N} X_i$. Y_l is the queue length difference between any two second stage VOQs. Then the average RD is

$$E\{RD\} \le E\{(\max Y_l)^+\},\ l = 1, 2, \ldots, N-1, \tag{14}$$

where $(X)^+ = \max\{0, X\}$.

According to central limit theorem, Y_l can be approximated by a normal distribution with mean $\eta = 0$, variance $\sigma^2 = 2Np(1-p) = 2\lambda[1 - \lambda/N(1-\lambda/N)]$, and

$$P(Y_l = k) \cong \frac{1}{\sigma\sqrt{2\pi}} e^{-k^2/2\sigma^2}, \quad k = -N, \ldots, 0, \ldots, N.$$

REFERENCES

[1] C. Chang, W. Chen, and H. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. IEEE INFOCOM*, 2000.

[2] C. Chang, D. Lee, and Y. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *Computer Commun.*, vol. 25, pp. 611-622, 2002.

[3] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, Aug. 1999.

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

[5] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches," in *Proc. IEEE INFOCOM*, NY, 2002, pp. 1024-1031.

[6] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proc. IEEE INFOCOM 1998*, vol. 2, NY, 1998, pp. 533-539.

[7] P. Giaccone, B. Prabhakar, and D. Shah, "Toward simple, high-performance schedulers for high-aggregate bandwidth switches," in *Proc. IEEE INFOCOM*, NY, 2002.

[8] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, pp. 188-201, Apr. 1999.

[9] H. J. Chao and J. S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch," in *Proc. IEEE ATM Workshop*, Fairfax, VA, May 1998.

[10] Y. Li, S. Panwar, and H. J. Chao, "On the performance of a dual round-robin switch," in *Proc. IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001.

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

[12] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12, Dec. 1999.

[13] P. Krishna, N. S. Patel, A.Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," in *Proc. IWQOS'98*, May 1998.
[14] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *Proc. IEEE INFOCOM*, vol. 2, NY, 2002, pp. 1032-1041.
[15] C. Chang, D. Lee, and Y. J. Shih, "Mailbox switch: a scalable two-stage switch architecture for conflict resolution of ordered packets," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004.
[16] C. Chang, D. Lee, and Y. Jou, "Load balanced Birkhoff-von Neumann switches, part II: multi-stage buffering," *Computer Commun.*, vol. 25, pp. 623-634, 2002.
[17] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.
[18] C.-Y. Tu, C.-S. Chang, D.-S. Lee, and C.-T. Chiu, "Design a simple and high performance switch using a two-stage architecture," in *Proc. IEEE Globecom*, 2005.
[19] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics (extended version)," Stanford University, Tech. Rep. TR03-HPNG-080101.
[20] N. Gogate and S.S.Panwar, "Assigning customers to two parallel servers with resequencing," *IEEE Commun. Lett.*, vol. 3, no. 4, pp. 119-122, Apr. 1999.
[21] Y. Li, S. S. Panwar, and H. Chao, "Exhaustive service matching algorithms for input queued switches," in *Proc. IEEE Workshop High Performance Switching Routing*, 2004.
[22] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Trans. Networking*, vol. 2, pp. 1-15, 1994.
[23] M. Schwartz, *Broadband Integrated Networks*. Prentice Hall, 1996.
[24] C. Chang, *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000.

Yanming Shen Yanming Shen is an Associate Professor in the Computer Science and Engineering Department at Dalian University of Technology, China. He received the Ph.D. degree from Department of Electrical and Computer Engineering at the Polytechnic University (now Polytechnic Institute of NYU) and his B.S degree in Automation from Tsinghua University, Beijing, P.R.China in 2000. He was a summer intern with Avaya Labs in 2006, conducting research on IP telephony. His general research interests include packet switch design, peer-to-peer video streaming, and algorithm design, analysis and optimization.

Shivendra S. Panwar Shivendra S. Panwar is a Professor in the Electrical and Computer Engineering Department at the Polytechnic Institute of New York University. He received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1981, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1983 and 1986, respectively.

He joined the Department of Electrical Engineering at the Polytechnic Institute of New York, Brooklyn (now Polytechnic Institute of NYU). He is currently the Director of the New York State Center for Advanced Technology in Telecommunications (CATT). He spent the summer of 1987 as a Visiting Scientist at the IBM T.J. Watson Research Center, Yorktown Heights, NY, and has been a Consultant to AT&T Bell Laboratories, Holmdel, NJ. His research interests include the performance analysis and design of networks. Current work includes cooperative wireless networks, switch performance and multimedia transport over networks.

He has served as the Secretary of the Technical Affairs Council of the IEEE Communications Society. He is a co-editor of two books, *Network Management and Control, Vol. II*, and *Multimedia Communications and Video Coding*, both published by Plenum, and has co-authored *TCP/IP Essentials: A Lab based Approach*, published by the Cambridge University Press. He was awarded, along with Shiwen Mao, Shunan Lin and Yao Wang, the IEEE Communication Society's Leonard G. Abraham Prize in the Field of Communication Systems for 2004.

H. Jonathan Chao H. Jonathan Chao (Fellow, IEEE, 2001) is Department Head and Professor of Electrical and Computer Engineering at Polytechnic University, New York, NY, where he joined in January 1992. He has been doing research in the areas of terabit switches/routers, network security, and quality of service control in high-speed networks. He holds 28 patents and has published over 150 journal and conference papers. He has also served as a consultant for various companies, such as Huawei, Lucent, NEC, and Telcordia.

During 2000-2001, he was Co-Founder and CTO of Coree Networks, NJ, where he led a team to implement a multi-terabit MPLS (Multi-Protocol Label Switching) switch router with carrier-class reliability. From 1985 to 1992, he was a Member of Technical Staff at Telcordia, where he was involved in transport and switching system architecture designs and ASIC implementations, such as the world's first SONET-like Framer chip, ATM Layer chip, Sequencer chip (the first chip handling packet scheduling), and ATM switch chip. From 1977 to 1981, he was a Senior Engineer at Telecommunication Labs of Taiwan performing circuit designs for a digital telephone switching system.

Prof. Chao is a Fellow of the IEEE for his contributions to the architecture and application of VLSI circuits in high-speed packet networks. He received the Telcordia Excellence Award in 1987. He is a co-recipient of the 2001 Best Paper Award from the IEEE Transaction on Circuits and Systems for Video Technology. He coauthored three networking books, Broadband Packet Switching Technologies - A Practical Guide to ATM Switches and IP Routers (New York: Wiley, 2001), Quality of Service Control in High-Speed Networks (New York: Wiley, 2001), and High-Performance Switches and Routers (New York: Wiley, 2007).

He has served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) on the special topics of "Advances in ATM switching systems for B-ISDN" (June 1997), "Next generation IP switches and routers" (June 1999), two issues on "High-performance optical/electronic switches/routers for high-speed Internet" (May and September 2003), and "High-speed network security" (Oct. 2006). He also served as an Editor for IEEE/ACM TRANSACTIONS ON NETWORKING from 1997-2000.

Prof. Chao received his B.S. and M.S. degrees in electrical engineering from National Chiao Tung University, Taiwan, and his Ph.D. degree in electrical engineering from Ohio State University.