Byte-Focal: A Practical Load Balanced Switch

Yanming Shen, Shi Jiang, Shivendra S. Panwar, H. Jonathan Chao
Department of Electrical and Computer Engineering
Polytechnic University, Brooklyn, NY 11201

Abstract— **Recently, a novel switch architecture, the load balanced (LB) switch proposed by C.S. Chang et al [1], [2] opened a new avenue for designing a large-capacity packet switch. The load balanced switch consists of two stages. First, a load-balancing stage spreads arriving packets equally among all linecards. Then, a forwarding stage transfers packets from the linecards to their final output destination. The load balanced switch does not need any centralized scheduler and can achieve 100% throughput under a broad class of traffic distributions. However, the load balanced switch 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. However, 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.**

I. INTRODUCTION

Internet traffic continues to grow rapidly. To keep pace with the demand, there has been significant research effort on designing high-speed large-capacity packet switch architectures that perform much better than 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 options, 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]–[6]). 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 [7]–[10]), 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 [11], [12] 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.

Recently, C.S. Chang et al introduced the Load Balanced Birkhoff-von Neumann switch architecture (LB-BvN) [1], [2]. This architecture is based on spreading packets uniformly 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. Therefore, load balanced switches are not subject to 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.

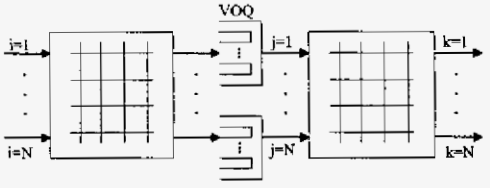

Fig. 1: The architecture of the load balanced Birkhoff-von Neumann switch

As shown in Figure 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. A rigorous proof of the 100% throughput result for the LB-BvN switch and the conditions on input traffic distribution are given in [2].

However, the FIFO discipline might be violated for packets from the same input 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) [13], Mailbox switch [14]). The second approach is to limit the packet 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) [15], EDF (Earliest Deadline First) [15], and FOFF (Full Ordered Frames First) [16].

In this paper, we propose a practical yet better performing

0-7803-8924-7/05/$20.00 (c)2005 IEEE.

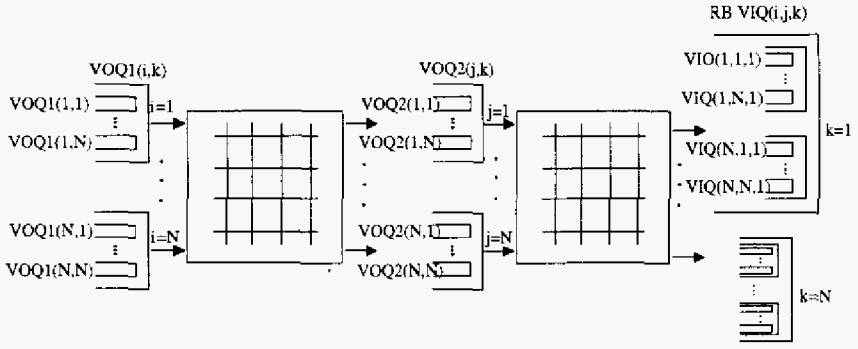

Fig. 2: The Byte-Focal switch architecture

switch architecture, the "Byte-Focal" switch, which uses a resequencing buffer to solve the out-of-sequence problem. We call the switch "Byte-Focal" to reflect the fact that packets of 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 presents the Byte-Focal switch architecture. In Section III, we propose various scheduling schemes at the first stage. The variable length packet delay performance is presented in Section IV, and Section V concludes the paper.

II. 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. Figure 2 shows the Byte-Focal switch architecture. It consists of two deterministic switch fabrics and three stages of queues, namely input queues i, center stage queues j, and output resequencing buffers (RB) k, where $i, j, k = 1, 2, \ldots, N$. The deterministic switch fabrics operate the same way as the basic LB-BvN switch, where both stages 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

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

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

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. As shown in Figure 3, 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

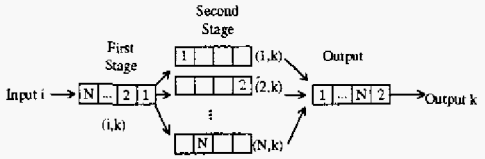

Fig. 3: Example flow of packets from VOQ1(i, k) through the switch

sent to the N second stage input ports in a round-robin manner and are placed in VOQ2$(1, k)$, VOQ2$(2, k)$, $\ldots$, VOQ2(N, k) according to their final destination. As a consequence of its mode of operation, 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 (see Figure 3 for an example).

The Byte-Focal switch uses the virtual input queue (VIQ) structure for the resequencing buffer(RB). 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 logical queues with each queue corresponding to a second stage input j. VIQ(i, j, k) separates each flow not only by its input port i, 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 obvious 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 queue j. 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 at the output, it is served, and the pointer moves to the next HOF packet location VIQ$(i, (j + 1)\ mod\ N, k)$.

Since there are N flows per output, more than one HOF

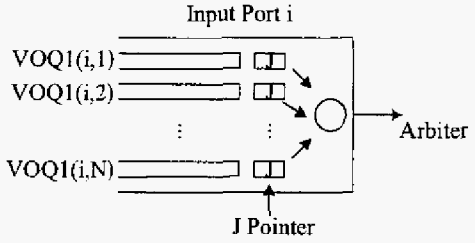

Fig. 4: Scheduling schemes at the first stage

packet may be eligible for service in a given 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 the round-robin service discipline. The DQ is simply a FIFO logical queue. It stores the indices of the VIQ sets, but only one from each VIQ set. When the HOF packet of VIQ set i arrives, index i joins the tail of the DQ. When a packet departs from the DQ, its index is removed if its next HOF packet has not arrived, and joins the tail of the DQ if its next HOF packet is at the output. 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 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.

III. FIRST STAGE SCHEDULING

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, more than one of the VOQ1s at i may be eligible to send packets to j. As shown in Figure 4, this problem can be stated as follows:

> *Each VOQ1(i,k) has a J pointer that keeps track of the last second stage input to which a packet was transferred, and the next packet is always sent to the next second stage input. As an HOL packet departs from a VOQ1(i,k), its J pointer value increases by one mod N. 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 needs to select one of them to serve.*

As we can see, the Byte-Focal switch performs the 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 $S_j(t) = \{\text{VOQ1}(i,k) | P_{ik}(t) = j\}$, then $S_j(t)$ is the set of VOQ1s that can send packets to the second stage input j at time t. We will next consider four ways of picking a VOQ1 to serve from the set $S_j(t)$.

A. Round-robin

To achieve small delay while maintaining fairness among all traffic flows, an efficient arbitration is necessary to schedule the departure of the HOL packets of the VOQ1s. One simple way to do the first stage scheduling is to use the **round-robin** scheme. In round-robin arbitration, among the set $S_j(t)$, the arbiter at each input port selects one of them in round-robin order. This scheme is simple and easy to implement.

B. 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 Figure 6). 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 longest queue from the set $S_j(t)$.

C. Fixed threshold scheme

The longest queue first scheme can achieve good performance. However, finding the longest queue can be time-consuming and is not practical for high-speed large-scale switches. It is easier to identify the congested VOQ1s by observing if their queue length exceeds a predetermined threshold (TH), N. Let $q_{ik}(t)$ be the length of the VOQ1(i,k), and $q_{is}(t)$ be the length of the VOQ1(i,s) being served. Define a subset $S_j'(t) = \{\text{VOQ1}(i,k) | \text{VOQ1}(i,k) \in S_j(t) \text{ and } q_{ik}(t) \geq TH\}$, 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:

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

D. Dynamic threshold scheme

As the switch size becomes large, setting the threshold to the switch size N causes large average delays. The reason is that the VOQ1 length has to reach a large value (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 $\lfloor Q(t)/N \rfloor$, where $Q(t)$ is the total VOQ1 queue length at an input port at time t. $\lfloor Q(t)/N \rfloor$ is therefore close to 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.

The proof of the following theorem appears in the Appendix.

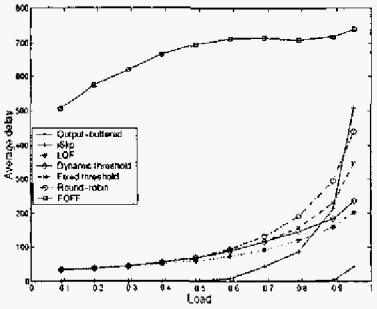

Fig. 5: Average delay under uniform traffic

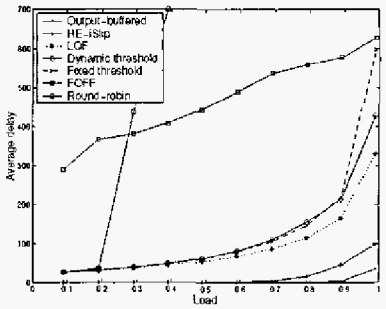

Fig. 6: Average delay under hot-spot loading

Theorem 1: The longest queue first, the fixed and dynamic threshold schemes are stable for any input traffic.

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 $\rho \in (0, 1)$. We use the following traffic scenarios to test the performance of the Byte-Focal switch:

Uniform i.i.d: $\lambda_{ij} = \rho/N$

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

Hot-spot: $\lambda_{ii} = \rho/2, \lambda_{ij} = \rho/2(N-1)$, for $i \neq j$. This type of traffic is more balanced than diagonal traffic, but obviously more unbalanced than uniform traffic.

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 Figure 5, 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. Compared to a single stage algorithm, iSlip, when the loading is low, iSlip has a smaller average delay, 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.

Figure 6 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, the throughput is only about 30%). For reference, we have also provided the performance of a typical single stage switch, HE-iSlip [9], [17] and the ouput-buffered switch. The LQF scheme has the best delay performance among the Byte-Focal switch schemes, but, unlike

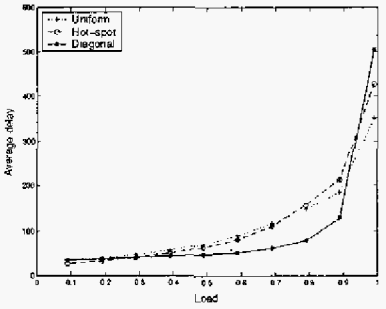

Fig. 7: Average delay for the dynamic threshold scheme

the fixed and dynamic threshold schemes, it is not practical due to its high implementation complexity. From Figure 6, 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 non-uniform 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 Figure 7, 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. The diagonal loading is very skewed and difficult to schedule using the centralized scheduling architecture. We also tried the Logdiagonal traffic matrix [6], 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 the traffic engineering design.

Figure 8 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 linearly dependent on the switch size. Since the Byte-Focal switch does not use a centralized scheduler, it can scale well, and can achieve good performance even for very large switch sizes.

A cell in the Byte-Focal switch experiences queueing delays

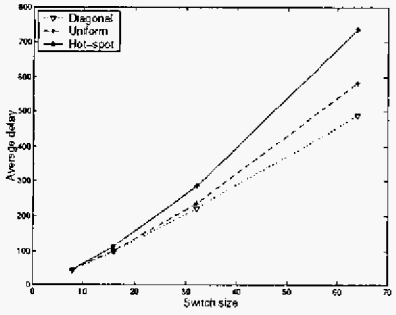

Fig. 8: Average delay vs. switch size N

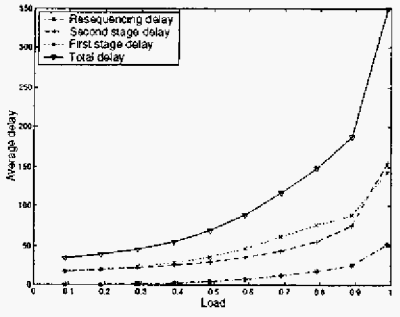

Fig. 9: 3-stage delays under uniform traffic for the dynamic threshold scheme

at the first stage and second stage, and resequencing delay at the output. Figure 9 shows the three components of the total delay. As seen, the first stage queueing delay and the second stage queueing delay are comparable, and the resequencing delay is much smaller compared to the other two delays.

Since Internet traffic is bursty [18], we also study the delay performance under bursty traffic. Consider the same simulation settings, but now the packets arrive in bursts. The 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 $\rho = \frac{10\mu}{1+9\mu}$). 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, with a probability that is uniformly distributed over N output ports.

Figure 10 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, when the traffic load is high.

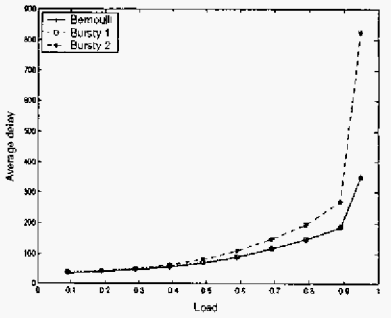

Fig. 10: Average delay of the dynamic threshold scheme under bursty traffic

IV. VARIABLE LENGTH PACKET DELAY PERFORMANCE

In the previous sections, only fixed length cell delay is considered. In this section, we will extend the delay analysis to variable length packet delay.

A. Variable length packet scheduling

We consider a switch designed to switch variable length packets, but internally switching fixed length cells. Packets are segmented into cells at input ports, transferred through the switching fabric, and reassembled at output ports. Since the resequencing delay and the reassembly delay overlap, the additional delay due to packet reassembly is reduced.

We consider two designs for the scheduling algorithm. The first one is cell mode scheduling, which is identical to the fixed length cell case. The second one is **packet mode scheduling**. In packet mode scheduling, the scheduling decision is performed at the packet boundary. Once we begin to serve a packet, we will keep serving it until the entire packet is transferred to the second stage (i.e., when a VOQ1 is enabled to transmit the first cell of a packet comprising k cells, it will be served at least for the following $k-1$ time slots).

Combining the packet mode scheduling and the dynamic threshold scheme, we have the following packet mode dynamic threshold algorithm:

1) At each time slot, if it is in the middle of a packet, keep serving this queue.
2) If not, apply the dynamic threshold scheme.

B. Worst case resequencing and reassembly delay

Compared to the fixed length cell delay, the variable length packets suffer additional reassembly delay at the output. In the Byte-Focal switch, the packet delay is the sum of queueing delays, and a combination of resequencing delay and reassembly delay. A simple bound for the sum of resequencing delay and reassembly delay follows.

Consider a packet comprising k cells. Assume among the cells of this packet, the first one arrives at the output at time t_0 (which is not necessarily the first cell of this packet), and this cell is denoted by C_1. The last one arrives at the output at time t_f (similarly, it is not necessarily the last cell of this packet), and it is denoted by C_2. Assume C_1 arrives at a second stage

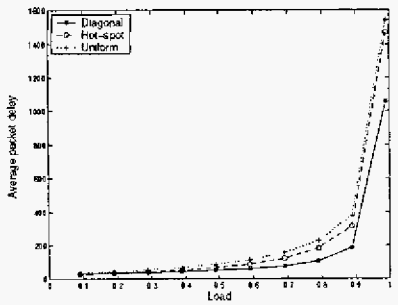

Fig. 11: Average packet delay of the dynamic threshold scheme with Internet-type packets

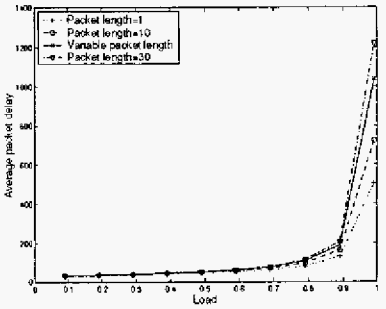

Fig. 12: Average packet delay of the dynamic threshold scheme for different packet length under diagonal loading

queue of size $q_1(t_1)$ at time t_1, and C_2 arrives at the second stage queue of size $q_2(t_2)$ at time t_2. It is easy to see that $t_0 = t_1 + Nq_1(t_1)$ and $t_f = t_2 + Nq_2(t_2)$.

$$\begin{aligned} t_f - t_0 &= t_2 + Nq_2(t_2) - t_1 - Nq_1(t_1) \\ &= t_2 - t_1 + N[q_2(t_2) - q_2(t_1) + q_2(t_1)] - Nq_1(t_1) \\ &= (t_2 - t_1) + N[q_2(t_2) - q_2(t_1)] + N[q_2(t_1) - q_1(t_1)] \end{aligned} \quad (3)$$

Due to the first stage scheduling algorithm, we have $t_2 - t_1 \leq k - 1$, and $q_2(t_2) - q_2(t_1) \leq k - 1$. We also have $|q_2(t_1) - q_1(t_1)| \leq N$ (Lemma 5 in [19]). Therefore,

$$t_f - t_0 \leq N^2 + (k-1) + N(k-1) \quad (4)$$

Note that the maximum resequencing delay for a cell is N^2 (which can be derived by setting $k = 1$). Furthermore, if the maximum packet length is k_{max} cells, then the sum of the resequencing delay and the reassembly delay is upper bounded by $N^2 + (k_{max} - 1) + N(k_{max} - 1)$.

C. Average packet delay performance

We report our simulation results for packet delay in this section, with the same simulation settings as before. The packet length is either fixed or has a trimodal distribution. The trimodal distribution, which approximates the observed Internet packet length distribution, is modeled as follows: an incoming packet has length 1 with a probability of 60%, 10 with a probability of 20%, and 30 with a probability of 20%.

Figure 11 shows the average packet delay with the trimodal packet length distribution. As with the cell delay, when the input traffic is non-uniform, the delay performance is not degraded, and even performs better. The reason is that when traffic is non-uniform, some queues will have more cells arriving than others, and the dynamic threshold scheme will serve these queues for more time slots, reducing switch over time and thus leading to less bandwidth wastage.

Generally, packet delays will increase as the packet length becomes larger, or for more variable length packets. For the trimodal packet length distribution, the average packet length is about 8.6. In Figure 12, we can see that the average delay increases with packet length and variability, but this is only a weak dependence.

To conclude, the simulation results show weak delay dependence on packet length, packet length distribution and the traffic matrix as long as the loading is kept constant. This simplifies traffic engineering for the Byte-Focal switch.

V. 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 per 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, packet lengths, and packet length distributions.

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

ACKNOWLEDGMENT

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).

REFERENCES

[1] C.S. Chang, W.J. Chen, and H.Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proceedings of IEEE INFOCOM*, 2000.
[2] C.S. Chang, D. Lee, and Y. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *Computer Communications*, 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. Communications*, vol. 47, no. 8, pp. 1260–1267, Aug 1999.
[4] D. Shah and M. Kopikare, "Delay bounds for approximate maximum weight matching algorithms for input queued switches," in *Proceedings of IEEE INFOCOM*, New York, 2002, pp. 1024–1031.
[5] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of IEEE INFOCOM 1998*, New York, 1998, vol. 2, pp. 533–539.
[6] P. Giaccone, B. Prabhakar, and D. Shah, "Toward simple, high-performance schedulers for high-aggregate bandwidth switches," in *Proceedings of IEEE INFOCOM*, New York, 2002.
[7] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, pp. 188–201, April 1999.
[8] H. J. Chao and J. S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch," in *Proceedings of IEEE ATM Workshop*, Fairfax, VA, May 1998.
[9] Y. Li, S. S. Panwar, and H.J. Chao, "Exhaustive service matching algorithms for input queued switches," in *Proceedings of IEEE Workshop on High Performance Switching and Routing*, 2004.
[10] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. on Computer Systems*, vol. 11, no. 4, pp. 319–352, Nov 1993.
[11] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12, December 1999.
[12] P. Krishna, N. S. Patel, A.Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," in *Proceedings of IWQOS'98*, May 1998.
[13] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *Proceedings of IEEE INFOCOM*, New York, 2002, vol. 2, pp. 1032–1041.
[14] C.S. Chang, D. Lee, and Y. J. Shih, "Mailbox switch: A scalable two-stage switch architecture for conflict resolution of ordered packets," in *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.
[15] C.S. Chang, D. Lee, and Y. Jou, "Load balanced Birkhoff-von Neumann switches, part II: multi-stage buffering," *Computer Communications*, vol. 25, pp. 623–634, 2002.
[16] I. Keslassy, S.T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, Aug 2003.
[17] Y. Li, *Design and Analysis of Scheduling for High Speed Input Queued Switches*, Ph.D. thesis, Polytechnic Universtiy, 2004.
[18] W. Willinger W. E. Leland, M. S. Taqqu and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 1–15.
[19] I. Keslassy, S.T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics (extended version)," Tech. Rep. TR03-HPNG-080101, Stanford University.

APPENDIX

In this section, we will show that the longest queue first, fixed threshold and dynamic threshold schemes can achieve 100% throughput. We will show that the total queue length at an input port is bounded, and therefore we only need a finite buffer at the first stage.

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

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.

Lemma 1: If $Q_i(t') \geq N(N-1)+1$ and $q_{is}(t') < N$, then it takes at most $N-1$ time slots to find a VOQ1(i,k) with $q_{ik}(t) \geq N$ to serve.

Proof: Since $Q_i(t') \geq N(N-1)+1$, and $q_{is}(t') < N$, therefore there exists a VOQ1(i,k), $k \neq s$, with queue length $q_{ik}(t') \geq N$. Assume the pointer at VOQ1(i,k) points to the second stage input j, and after T time slots (obviously, $T \leq N-1$), input i is connected to j. At time $t'+T$, we have

$$q_{ik}(t'+T) \geq q_{ik}(t') \geq N,$$

which means that at time $t'+T$, in set $S_j(t'+T)$, there is a queue with queue length greater than N.
For LQF, the longest queue is chosen from set $S_j(t'+T)$, therefore, $q_{is}(t'+T) \geq N$. For the fixed threshold scheme, $S_j'(t'+T)$ is nonempty, therefore, $q_{is}(t'+T) \geq Th = N$. For the dynamic threshold scheme, at time t', there exists a VOQ1(i,k) with queue length $q_{ik}(t') \geq Th(t')+1 = \left\lfloor \frac{Q_i(t')}{N} \right\rfloor + 1 \geq N$. At time $t'+T$, $Th(t'+T) = \left\lfloor \frac{Q_i(t'+T)}{N} \right\rfloor \leq \left\lceil \frac{Q_i(t')+N-1}{N} \right\rceil \leq Th(t')+1$, and we have $q_{ik}(t'+T) \geq Th(t'+T)$, and $S_j'(t'+T)$ is nonempty, therefore, $q_{is}(t'+T) \geq N$.

Thus for LQF, fixed threshold and dynamic threshold schemes, we always have $q_{is}(t'+T) \geq N$ with $T \leq N-1$. ■

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

Proof: 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 which is from $q_{is}(t_b) = N-1$ to $q_{is}(t_e) = N$, which is the duration that the queue being served stays below N. From Lemma 1, we have $t_e - t_b \leq N-1$. Within each cycle, we will show there is no time slot wasted. At the beginning t_b of each cycle, $q_{is}(t_b) = N-1$, and at the end t_e of each cycle, $q_{is}(t_e) = N$. Since there is at most one departure in a time slot, then for any $t_b \leq t < t_e$,

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

according to any of the service policies used. Since $t < t_e$, we have

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

But $t_e - t_b \leq N-1$, thus $q_{is}(t) > 0$ and there are no time slots wasted. We therefore conclude that the first stage is work-conserving. ■

From Lemma 1 and Lemma 2, we can prove theorem 1.

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

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

Thus, by the definition of stability, the system is stable. ■