The Interleaved Matching Switch Architecture

Bill Lin, *Member, IEEE*, and Isaac Keslassy, *Member, IEEE.*

***Abstract*—Operators need routers to provide service guarantees such as guaranteed flow rates and fairness among flows, so as to support traffic engineering and real-time traffic. However, current centralized input-queued router architectures cannot scale to fast line rates while providing these service guarantees. On the other hand, while load-balanced switch architectures that rely on two identical stages of fixed-configuration switches appear to be an effective way to scale Internet routers to very high capacities, there is currently no practical and scalable solution for providing service guarantees in these architectures.**

In this paper, we introduce the interleaved matching switch (IMS) architecture, which relies on a novel approach to provide service guarantees using load-balanced switches. The approach is based on emulating a Birkhoff-von Neumann switch with a load-balanced switch architecture and is applicable to any known admissible traffic. We show that service guarantees, 100% throughput, and packet ordering can be achieved with $O(1)$ online complexity. In cases where fixed frame sizes are applicable, we also present an efficient offline frame-based decomposition method. More generally, we show that the IMS architecture can be used to emulate any input queued or combined input-output queued switch, leveraging a large body of known results for ensuring stability.

***Index Terms*—Load-balanced routers, high-performance switches, rate guarantees, switch emulation.**

I. Introduction

A. Background

THERE has been much interest recently in a class of switch architectures called *load-balanced routers* [1]–[7], [9]–[15]. This class of architectures is based on a *load-balanced switch architecture* where two identical stages of fixed configuration switches are used for routing packets. Figure 1 shows a diagram of a generic two-stage load-balanced switch architecture (all figures are in the appendix). The first switch connects the first stage of input linecards to the center stage of intermediate input linecards, and the second switch connects the center stage of intermediate input linecards to the final stage of output linecards. As shown in [4], this class of architectures appears to be a practical way to scale Internet routers to very high capacities and line rates. The scalability of this class of architectures can be attributed to two key aspects. First, they do not require a scheduler. Second, these architectures are built using two identical stages of fixed configuration switches whose deterministic interconnection patterns are independent of packet arrivals. Thus, there is no need for arbitrary per-packet dynamic switch configurations, which are extremely difficult to achieve at high-speeds. The use of fixed configuration switches is particularly amenable to scalable implementations with optics, as exemplified by the 100 Tb/s reference design described in [4]. This reference design is based on a fixed hierarchical mesh of optical channels that interconnects $N = 640$ linecards, each operating at a rate of $R = 160$ Gb/s.

Although the load-balanced routers described in [1]–[7], [9]–[11] can achieve guaranteed throughput for all admissible traffic on a *best effort* basis, they do not provide *service guarantees* required by network operators to support real-time traffic, bandwidth provisioning, and various other critical traffic engineering tasks. In particular, backlogged flows cannot obtain any service rate guarantees, and therefore flows could easily get starved or suffer from extreme service unfairness.

On the other hand, as shown in [16]–[20], centralized input-queued router architectures are able to provide such service guarantees, by using a Birkhoff-Von Neumann scheduling algorithm [21], [22]. Nevertheless, centralized input-queued router architectures cannot easily scale to higher line rates and port counts. Therefore, the objective of this paper will be to investigate an approach for providing service guarantees that retains the key scaling properties of the load-balanced switch architecture.

In [12], two alternative schemes are proposed for providing service guarantees on a load-balanced switch. The first scheme is based on timestamping individual packets and buffering packets at the center stage. Packets are then scheduled at the center stage using an Earliest-Deadline-First (EDF) scheduling policy. Using an EDF scheduling policy, packets may arrive at the final destination output linecard out-of-order. A packet resequencing mechanism is implemented at the final output stage to correct mis-sequenced packets. As noted in [12], the proposed EDF-scheduling and resequencing mechanisms require complex hardware that are difficult to implement at very high speeds. The second scheme is a frame-based approach that requires choosing a frame size T. The end-to-end delay of this approach is $\Theta(NT)$, where N is the number of switch ports. Therefore, a large frame size implies a large multiplicative end-to-end packet delay. On the other hand, a small frame size implies a large rate granularity. As a result, the approach fails to provide uniform service guarantees for all non-uniform traffic patterns.

B. Contributions of the Paper

The main contributions of this paper are as follows:

Paper approved by T. T. Lee, the Editor for Switching Architecture Performance of the IEEE Communications Society. Manuscript received July 26, 2008; revised January 16, 2009 and March 26, 2009.

B. Lin is with the University of California, San Diego, La Jolla, CA 92093–0407 (e-mail: billlin@ece.ucsd.edu).

I. Keslassy is with the Technion – Israel Institute of Technology, Haifa 32000, Israel (e-mail: isaac@ee.technion.ac.il).

This work was supported in part by the ATS-WD Career Development Chair.

This work was presented in part at the 13th Annual IEEE Symposium on High-Performance Interconnects (HotI), Stanford, CA, August 2005 [8].

Digital Object Identifier 10.1109/TCOMM.2009.080380

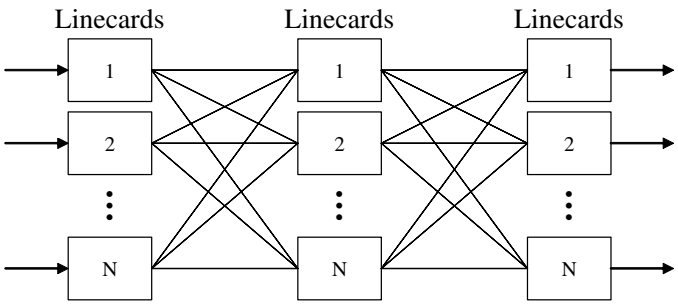

Fig. 1: Generic load-balanced switch architecture.

- First, from a theoretical point of view, we show that our proposed interleaved matching switch (IMS) architecture can be used to emulate *any* input queued (IQ) or combined input-output queued (CIOQ) switch by *interleaving* the associated matchings across the intermediate input nodes of a load-balanced switch. In particular, we prove that the departure time of every packet on a load-balanced switch architecture is within a *constant delay* from a corresponding shadow IQ or CIOQ switch under the same matchings. Consequently, many of the throughput and service guarantees provided in the literature for IQ and CIOQ switches directly extend to load-balanced switches.
- Second, in the case when the traffic is known a priori, we show that our proposed IMS architecture can practically provide service guarantees by *emulating* a Birkhoff-von Neumann input-queued switch [19]. In addition to using scalable fixed configuration switches, an IMS only requires a fully-distributed online fair queueing mechanism at each input linecard, based only on local state information. This online scheduling can be achieved in $O(1)$ complexity by using either a round-robin based scheduler [23], [24] or a timestamp-based scheduler with a pipelined priority queue mechanism [25], [26]. Since packet ordering is enforced throughout the switch, the IMS architecture does not require packet resequencing at the output or the associated resequencing delays. IMS can provide uniform service guarantees for any admissible traffic where the traffic profile is known.
- Finally, in cases where a fixed frame size is applicable, we present an efficient offline decomposition method that has significantly lower complexity than Birkhoff-von Neumann decomposition. In contrast to the frame-based scheme presented in [12] that has an $\Theta(NT)$ end-to-end delay bound, we show that our approach has a significantly lower $\Theta(T + N)$ end-to-end delay bound. For instance, when T is a constant multiple of N, the frame-based scheme presented in [12] incurs an $\Theta(N^2)$ end-to-end delay bound, whereas our approach only incurs an $\Theta(N)$ end-to-end delay bound. The frame-based decomposition method is also applicable to the case of a single crossbar switch.

C. Organization of the Paper

The rest of the paper is organized as follows. Section II introduces the IMS architecture. Section III demonstrates that the IMS architecture can emulate any IQ or CIOQ switch, leveraging known results for achieving stability and service guarantees. In particular, Section IV shows that service guarantees can be provided by emulating a Birkhoff-von Neumann switch and that the emulation can be implemented with constant time online complexity in a fully distributed manner. Section V presents a more efficient offline decomposition algorithm than Birkhoff-von Neumann decomposition for the case where a fixed frame size is applicable. This frame-based decomposition scheme for the IMS architecture is shown to achieve low end-to-end delay. Section VI presents an illustrative example and experimental results. Finally, Section VII presents conclusions.

II. The Interleaved Matching Switch

This section describes the IMS architecture. The basic idea of the IMS architecture is to emulate any IQ or CIOQ switch over two identical stages of fixed configuration switches by using the same matching algorithm m. In particular, we will prove later in Section III that the departure time of every packet on an IMS is within a constant delay from a corresponding shadow IQ or CIOQ switch under the same matchings. Therefore, many of the throughput and service guarantees provided in the literature for IQ and CIOQ switches directly extend to the IMS architecture. The key benefit of emulating an IQ or a CIOQ switch over fixed configuration switches is the ability to scale the switch fabric to very high switch capacities and line rates. We defer to Sections III, IV, and V for detailed discussions on the emulation properties of IMS.

In this section, we first present an overview of the IMS architecture. We then present in more details how a packet is routed through the IMS architecture. Finally, we describe delay optimizations that can reduce the constant for the fixed propagation delay through the switch.

Note that throughout this paper, we assume that packets have a fixed length and time is slotted.

A. Overview of the Architecture

The IMS architecture consists of three linecard stages that are interconnected by two fixed configuration switches, exactly like the load-balanced switch architectures described in [1], [2], [4]. However, in case of congestion, these architectures primarily buffer packets in the *center* stage whereas the IMS architecture primarily buffers packets in the *input* stage. This is depicted in Figure 2.

Specifically, the first stage consists of N input linecards. Each input linecard i maintains N virtual output queues (VOQ) for buffering incoming packets, one per final output destination. The center stage consists of N intermediate input linecards. Each intermediate input linecard j maintains a set of N slots: $B_{j1} \ldots B_{jN}$. These slots are used for coordination, as we shall see in Section III. The final stage consists of N output linecards where packets depart.

To simplify presentation, we model the two switches as *uniform meshes*, as in [4]. Each linecard in the first stage is connected to each linecard in the center stage by a channel at rate R/N via the first mesh, where R is the line rate and N is the number of linecards. Similarly, each linecard

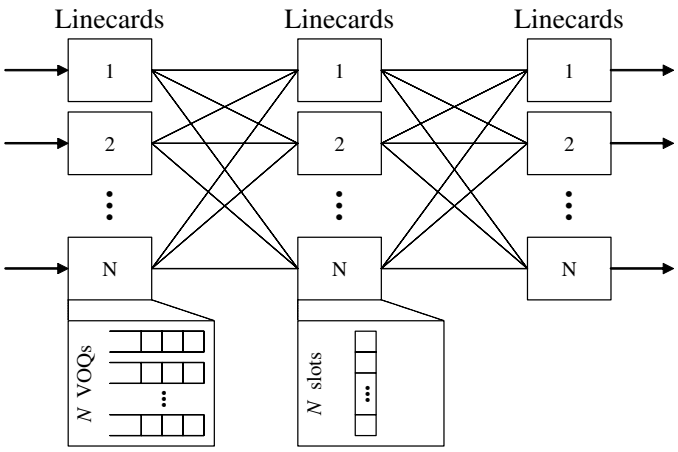

Fig. 2: The Interleaved Matching Switch Architecture.

in the center stage is connected to each linecard at the final stage by a fixed equal rate channel at rate R/N via the second mesh. As described in [4], the uniform mesh model can be readily implemented at very high capacities and line rates using different types of switches, such as optical meshes with space and/or wavelength multiplexing, as well as time-multiplexed cyclic permutation switches (also called round-robin switches) with no speed-up.

As usual, we assume that a packet sent at the start of time slot t through a line of rate R will be completely transferred by the end of time slot t, taking one full time slot. Similarly, a packet sent through a line of rate R/N will take N full time slots, and therefore will be transferred by the end of time slot $t + N - 1$.

B. Packet Path

To understand the operation of the IMS architecture, consider a stream of packets arriving at an input linecard i, one new packet at the start of every time slot t. The packets are immediately queued at their appropriate VOQs, depending on their final output destination.

At the start of every time slot t, after packet arrivals, each input linecard i selects a VOQ Z_{ik} to serve, using a selection mechanism described later. Then, all input linecards send in parallel the packets at the head of their respective selected VOQs to intermediate input linecard j, where

$$j = ((t-1) \mod N) + 1. \tag{1}$$

These packets can be sent in parallel over the first mesh because each linecard in the first stage is connected to each linecard in the center stage by a channel of rate R/N. Therefore, no speed-up is required.

As explained above, intermediate input linecard j then receives up to N packets in parallel by the end of time slot

$$t + N - 1. \tag{2}$$

These packets are stored in the set of slots $B_{j1} \dots B_{jN}$. Specifically, if a packet is destined for output k, it is stored in B_{jk}. To avoid conflict, we need to ensure that packets received from different inputs are destined to different outputs. This is the usual bipartite matching constraint, as we shall see.

Then, at the start of time slot

$$t + N, \tag{3}$$

intermediate input linecard j sends up to N packets in parallel over the second mesh to the N output linecards, including at most one packet to each output linecard k from the corresponding slot B_{jk}. Again, these packets can be sent in parallel over the second mesh with no speed-up. Each output linecard k then receives the packet sent by intermediate input linecard j by the end of time slot

$$(t + N) + (N - 1) = t + 2N - 1, \tag{4}$$

and the packet departs immediately from the router.

The above process operates continuously. Specifically, at time slot $t = 1$, all input linecards can send a packet each in parallel to intermediate input linecard $j = 1$, and all output linecards may receive a packet each in parallel from intermediate input linecard $j = 1$ by time slot $1 + 2N - 1$. At time slot $t = 2$, all input linecards can send packets to intermediate input linecard $j = 2$, and all output linecards may receive packets from intermediate input linecard $j = 2$ at time slot $2 + 2N - 1$, and so on. Thus, each input linecard can continuously select at every time slot a new packet to send over the first mesh, and each output linecard may continuously receive a new packet at every time slot from the second mesh for departure. All linecards operate in parallel, and the operations of the first mesh and the operations of the second mesh effectively overlap in time.

Note that the above operation implies that each input linecard sends packets in *round-robin* order to intermediate input linecards, and each output linecard receives packets in the same round-robin order from intermediate input linecards, starting with the first intermediate input linecard, moving next to the second intermediate input linecard, and so on, possibly not sending a packet to (or receiving a packet from) a particular intermediate input linecard at some time slots.

Also, as depicted in Figure 3, note that the two uniform meshes can be replaced by a single mesh running twice as fast, with each linecard containing three logical parts (input, intermediate input, and output). The channels in the first logical mesh can be *time-multiplexed* with the channels in the second logical mesh, thus reducing the total propagation delay.

For instance, linecard 2 sends traffic to linecard 3 in both the first and second stages, both at rate R/N. Therefore, by providing a link of total rate $2R/N$, we can time-multiplex the two channels into one, e.g. by dividing each time-slot into two sub-time-slots, each being devoted to a different stage. In Figure 3, we represent the three logical parts (input, intermediate input, and output) of linecard 2 using a single symbol. We then use the link linecard 2 $\rightarrow$ linecard 3 twice at rate R/N each, providing the total rate of $2R/N$.

III. IMS Can Emulate Any IQ or CIOQ Switch

In this section, we prove that the IMS architecture can emulate any IQ or CIOQ switch using the same matching algorithm m. There are two main reasons behind our desire to emulate IQ and CIOQ switches. The first reason is a

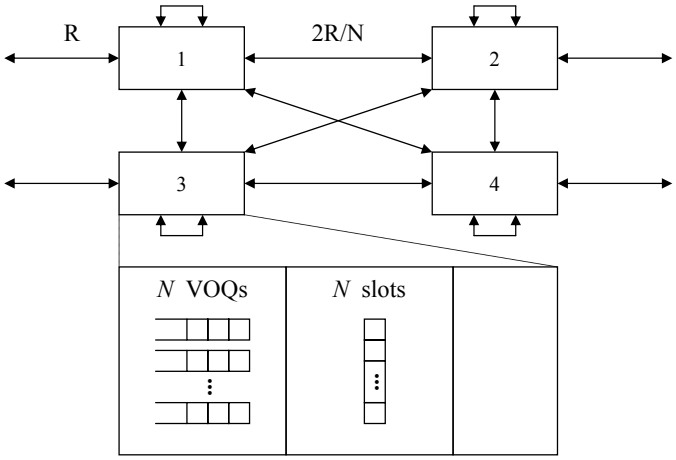

Fig. 3: The Interleaved Matching Switch architecture using a single combined mesh, shown with $N = 4$.

fundamental and historical need to better understand how *new* switch architectures work by studying how they would emulate *current* switch architectures. The second reason is the desire to emulate the Birkhoff-von Neumann input-queued switch [19], which is known to provide service guarantees with only a low-complexity online scheduler that can be fully distributed. The emulations of IQ and CIOQ switches are shown in Sections III-A and III-B, respectively.

A. Emulating Any IQ Switch

The intuition behind the emulation of any IQ switch by an IMS is as follows: every N time slots, an IQ switch consecutively makes N matchings between its inputs and its outputs, and transfers packets from the inputs to the outputs accordingly. An IMS will make the very same N matchings, and implement each matching using a different intermediate input linecard. Therefore, putting aside the fixed propagation delays, it will move the same packets at the same time, hence emulating the IQ switch. In the next paragraphs, we will first formalize this intuition by defining the terms used in the proof, and then we will prove that an IMS can emulate any IQ switch.

Definition 1 (Conflict-Free Matching): Let m be a matching algorithm, let $\pi^m(t)$ represent the matching of all inputs to outputs under the matching algorithm m at time slot t, and let $k = \pi^m(i,t)$ denote the unique output matched to input i under matching algorithm m at time slot t. Then $\pi^m(t)$ is said to be a conflict-free matching if and only if $i_1 \neq i_2 \Rightarrow \pi^m(i_1,t) \neq \pi^m(i_2,t)$.

In the remainder, we assume that an IQ switch only uses conflict-free matchings.

Definition 2 (Match Time): Let c be a packet queued at a VOQ Z_{ik} at input linecard i. Let $t_{match}(c)$ be the time slot that it gets matched for transfer from input linecard i. Then $t_{match}(c)$ is referred to as the match time for c.

Definition 3 (Departure Time): Let c be a packet that has been matched for transfer and it is destined to output linecard k. Let $t_{depart}(c)$ be the time slot that it completely arrives at its destination output linecard k. Then $t_{depart}(c)$ is referred to as the departure time of c. We assume that once a packet completely arrives at an output linecard, it departs immediately through its outgoing link.

Definition 4 (Shadow Switch): Let X be a switch. A shadow switch Y is a switch with the same number of input and output ports as X. It receives identical input traffic patterns, and operates at the same line rate as X.

Definition 5 (Emulation): A switch X is said to emulate a shadow switch Y if under identical inputs, the departure times for identical packets are within a constant.

In other words, two switches emulate each other if they have the same queueing delay under all possible traffic patterns (ignoring the fixed propagation delays inside the switches). The following theorem shows that an IMS can emulate any IQ switch. It is illustrated using a practical example of IQ emulation in Section VI-A.

Theorem 1 (IQ Emulation): An IMS can emulate any IQ switch under the same matching algorithm m.

Proof: Let Y be a shadow IQ switch that uses some conflict-free matching algorithm m. Let X be an IMS.

Assume that X uses the same matching algorithm m to select which VOQ Z_{ik} to service at every time slot. By assumption, the inputs in X as well as Y have the same arrival process, both switches use the same matching algorithm, and in both switches packets depart from the inputs as soon as they are matched. Therefore, by induction, we can see that at all time slots, both input stages have the same arrival and departure processes, as well as the same states. In particular, for every packet c scheduled in the shadow IQ switch Y, match time $t^X_{match}(c) = t^Y_{match}(c)$.

For a shadow IQ switch with a crossbar implementation, once a packet c has been scheduled, it is assumed to depart through the corresponding output in the same time slot. That is,

$$t^Y_{depart}(c) = t^Y_{match}(c). \tag{5}$$

For the IMS X, once a packet c has been scheduled, there are no conflicts in the middle linecards since m is conflict-free. Following Equation (4), the packet c will depart through the corresponding output in time slot

$$t^X_{depart}(c) = t^X_{match}(c) + 2N - 1. \tag{6}$$

Therefore, the difference in departure time is a constant delay (propagation time):

$$t^X_{depart}(c) - t^Y_{depart}(c) = 2N - 1. \tag{7}$$

■

In particular, this theorem applies to algorithms like the MWM (Maximum Weight Matching) scheduling algorithm, which is known to be strongly stable, and therefore provide 100% throughput [27], [28].

Corollary 1 (IMS-MWM): IMS provides 100% throughput when using MWM.

It is interesting to note that, in effect, the set of slots $B_{j1} \ldots B_{jN}$ at each intermediate input linecard j corresponds to an equivalent crossbar configuration in the shadow IQ switch at some point in time. In fact, this architecture can be seen as extending the idea of *time-slicing among parallel crossbars* to time-slicing among intermediate linecards. It is a particular case of time-space conversion.

At first glance, this architecture also shares similarities with a *Clos network*. However, there is a crucial difference in that a Clos network needs active (electronic) switch elements that require per-slot dynamic configuration for each match, whereas the current architecture is able to use passive (optical) elements.

B. Emulating Any CIOQ Switch

In this section, we extend the results in Section III-A to show that the IMS architecture can also emulate any CIOQ switch under the same positive integer *speedup* S (e.g. $S = 2$) and matching algorithm m. In a conventional CIOQ crossbar switch with a speedup of S, S matching phases are performed in every time slot. Corresponding to the S matching phases, up to S packets may be served from each input, and up to S packets may be received at each output. Each output maintains an output queue since only one packet may depart from it at each time slot, and the crossbar switch operates S times faster.

To emulate a CIOQ switch, we make several extensions to the IMS architecture. First, the two meshes operate at a speedup of S: each linecard in the first stage is connected to each linecard in the center stage by a channel at rate SR/N via the first mesh, and each linecard in the center stage is connected to each linecard in the final stage by a channel at rate SR/N via the second mesh.

Then, at every time slot n, each input linecard i performs S matching phases and selects S VOQs to service based on matching algorithm m, possibly selecting the same VOQ more than once.

Definition 6 (Match Time under Speedup): Let $\mathcal{C}$ be the set of up to S packets selected in the S matching phases at time slot n. Then for all $c \in \mathcal{C}$, the match time is $t_{match}(c) = n$.

At every time slot, an intermediate input linecard j is chosen in the same way as described in Section III-A (i.e., see Equation (1)). The S packets selected are then sent to this intermediate input linecard via a channel at rate SR/N through the first mesh, and will all completely arrive by the end of time slot $n + N - 1$.

Then, at each intermediate input linecard, instead of each B_{jk} holding one packet, B_{jk} can buffer up to S packets destined to output k.

Then, at the start of time slot $n+N$, each intermediate input linecard j may send up to S packets to each output linecard k from the packets buffered at B_{jk}. These packets will be sent to output linecard k via a channel at rate SR/N through the second mesh, which will completely arrive at outline linecard k for departure by the end of time slot $n + 2N - 1$. Since up to S packets may arrive at an output in a given time slot, and only one packet can depart through the outgoing link, they have to be queued in the output queue. Therefore, instead of defining a departure time, we define an output queue arrival time as follows.

Definition 7 (Output Queue Arrival Time): Let $\mathcal{O}$ be the set of up to S packets received at an output k in time slot t. Then for all $c \in \mathcal{O}$, the output queue arrival time is $t_{output}(c) = t$.

Note that this $t = n+2N-1$ is the same as in Equation (4) defined in Section II. Finally, since up to S packets may be received at an output each time slot, but only one can depart, output queues are added to each output linecard. With these extensions, the packets will arrive in order at their final output destinations as before.

Theorem 2 (CIOQ Emulation): An IMS can emulate any CIOQ switch under the same speedup S and matching algorithm m.

Proof: The proof follows the same line of arguments as Theorem 1. Let Y be a shadow CIOQ switch with speedup S and some conflict-free matching algorithm m. Let X be an IMS, and assume that X has the same speedup and uses the same matching algorithm m in S matching phases to select up to S packets to service at every time slot. By assumption, both switches have the same arrival process, both switches make the same matchings, and packets depart from inputs as soon as they are matched in both switches. By induction, both input stages have the same arrival and departure processes, and the same states. Therefore, for every packet c scheduled in Y, match time $t^X_{match}(c) = t^Y_{match}(c)$, and departure time at Y is $t^Y_{output}(c) = t^Y_{match}(c)$. For X, since there is a constant $2N - 1$ delay through the switch from match time to arrival at the destination output, $t^X_{output}(c) = t^X_{match}(c) + 2N - 1$. Therefore, the difference in arrival time to the destined output queue is also constant:

$$t^X_{output}(c) - t^Y_{output}(c) = 2N - 1. \quad (8)$$

Since only one packet can depart for an outgoing link in both switches, their output queue lengths when c arrives at their respective output queue will also be the same. Therefore, the $2N - 1$ difference in output queue arrival time will remain in the difference in departure time. ■

In particular, this theorem applies to *maximal* matching algorithms such as iSLIP [29], since they are known to be strongly stable and provide 100% throughput with speedup two [27].

Corollary 2 (IMS-Maximal): IMS provides 100% throughput when using any maximal matching algorithm and speedup two.

Further, using a speedup of two, there exist matching algorithms that can emulate output-queued switches [30]. Therefore, we obtain the following result.

Corollary 3 (OQ Emulation): IMS can emulate an output-queued switch with a speedup of two.

Finally, we can also use this theorem to obtain a *packet-mode IMS architecture*, in which cells belonging to the same packet are forced to arrive consecutively to their output destination. As shown in [31], this can be done for any speedup $S = 2 + \epsilon$ with $\epsilon > 0$.

Corollary 4 (Packet-Mode OQ Emulation): IMS can emulate a packet-mode output-queued switch with a speedup arbitrarily close to two.

IV. Providing Service Guarantees

In the previous section, we showed that the IMS architecture can emulate any IQ or CIOQ switch. In this section, we show that service guarantees can be achieved by emulating a Birkhoff-von Neumann input-queued switch, which is known to provide service guarantees when the traffic is known a

priori. This is described in Sections IV-A and IV-B. We show in Section IV-C that the emulation can be realized in a fully distributed manner using $O(1)$ online algorithms.

A. Background on Birkhoff-von Neumann Decomposition

In [19], an approach based on Birkhoff-von Neumann decomposition was presented for providing service guarantees on IQ crossbar switches when the traffic profile is known *a priori*. In particular, let $\Lambda = (\lambda_{ik})$ be an $N \times N$ non-negative arrival traffic rate matrix, where λ_{ik} represents the mean rate of traffic from input i to output k. $\Lambda = (\lambda_{ik})$ is said to be *admissible* and *doubly sub-stochastic* when

$$\forall k,\ \sum_{i=1}^{N} \lambda_{ik} \leq 1,\ \text{and}\ \ \forall i,\ \sum_{k=1}^{N} \lambda_{ik} \leq 1.$$

and it is said to be *doubly stochastic* if these are all equalities. Given an admissible traffic rate matrix Λ, the problem of crossbar matching can be defined as a decomposition of Λ into a series of permutation matrices (π_h) such that

$$\Lambda \leq \sum_{h=1}^{H} \phi_h \pi_h \tag{9}$$

where $0 < \phi_h \leq 1$ and $\sum_h \phi_h = 1$. With the decomposition, each permutation matrix π_h can then be used for crossbar matching for the corresponding fraction of time ϕ_h. Here, we overload the notation of π_h to represent both a permutation matrix and the corresponding matching that it implies.

The overall approach in [19] consists of the following:

1) It first uses an $O(N^3)$ von Neumann algorithm [22] to convert an admissible traffic rate matrix into a doubly stochastic matrix.
2) It then uses an $O(N^{4.5})$ Birkhoff decomposition algorithm [21] to find the decomposition shown in Equation (9). Both steps 1 and 2 are performed *offline*.
3) It finally uses the Packetized Generalized Process Sharing (PGPS) algorithm in [32], [33] to determine which permutation matrix π_h (obtained in Step 2) is to be used in a given time slot in proportion to ϕ_h. This *online* scheduling step has $O(\log N)$ complexity, but this complexity can be reduced to $O(1)$ time by using a pipelined priority queue mechanism [25], [26] or by replacing fair queueing with a round-robin based scheduler [23], [24].

B. Birkhoff-von Neumann Switch Emulation, Throughput Guarantees, and Service Guarantees

It follows directly from Section III-A, and in particular from Theorem 1, that an IMS can emulate a Birkhoff-von Neumann input-queued switch, and therefore provide the same throughput and service guarantees, as stated in the next theorem.

Theorem 3: An IMS can emulate a Birkhoff-von Neumann IQ switch.

Using this emulation theorem, we can directly extend most of the properties of a Birkhoff-von Neumann switch to an IMS that emulates it.

Theorem 4: If the arrival traffic is a stationary and ergodic stochastic process with the strictly doubly sub-stochastic mean rate $\Lambda = (\lambda_{ik})$, then an IMS provides 100% throughput.

Proof: This is a known result for the Birkhoff-von Neumann IQ switch [1], [19]. It extends directly to the IMS architecture using the emulation proved in Theorem 3. ■

We next show that an IMS has the same service guarantees as the Birkhoff-von Neumann switch that it emulates.

Theorem 5: Let $\mathcal{F}_{ik}$ be a continually backlogged flow from input i to output k, and let $C_{ik}(t)$ be the cumulative number of its served packets by time t. Let t_1 and t_2 be two time slots such that $(2N-1) \leq t_1 \leq t_2$. Then any time-independent bounds on service guarantees defined by

$$C_{ik}(t_2) - C_{ik}(t_1)$$

are exactly the same for an IMS and the input-queued switch that it emulates.

Proof: Let X be an IMS, and let Y be an input-queued switch that it emulates. Given that the departure time for a packet in X is a constant offset $(2N-1)$ from the departure time for the same packet in Y (Equation (4)), we have

$$C_{ik}^{X}(t_2) = C_{ik}^{Y}(t_2 - (2N-1)), \tag{10}$$

$$C_{ik}^{X}(t_1) = C_{ik}^{Y}(t_1 - (2N-1)). \tag{11}$$

Then it follows that any time-independent bounds defined by $C_{ik}(t_2) - C_{ik}(t_1)$ are the same for X and Y. ■

In [19], it was shown that for any admissible traffic rate matrix $\Lambda = (\lambda_{ik})$, the difference in the cumulative number of packets served from a continually backlogged flow $\mathcal{F}_{ik}$ in a Birkhoff-von Neumann switch for any $t_1 \leq t_2$ is bounded by

$$\sum_{h \in E_{ik}} \phi_h (t_2 - t_1) - \sigma_{ik} \leq C_{ik}(t_2) - C_{ik}(t_1) \leq \sum_{h \in E_{ik}} \phi_h (t_2 - t_1) + \sigma_{ik}, \tag{12}$$

where

$$\sigma_{ik} = \min \left[H, |E_{ik}| + \sum_{h \in E_{ik}} \phi_h (H-1) \right],$$

E_{ik} is the subset of permutation matrices with the (i,k) entry equal to 1, and H is the number of permutation matrices in Equation (9). Therefore, it follows that an IMS that emulates a Birkhoff-von Neumann switch provides the same service guarantees.

C. Optimizing the IMS Architecture for Service Guarantees

In this section, we describe a number of optimizations that can be used in emulating a Birkhoff-von Neumann switch.

1) *Distributed Scheduling*: Instead of performing PGPS scheduling in a centralized manner, each input linecard can perform the same PGPS scheduling in a *fully-distributed* manner to select which of its own VOQs to service at each time slot. Using a pipelined-based priority queue mechanism, this fair queueing step can be reduced to $O(1)$ complexity **[25], [26]**. Alternatively, another practical solution is to use an $O(1)$ complexity round-robin based scheduler [23], [24] to approximate

PGPS scheduling. The combination of the IMS architecture and fully distributed online scheduling with $O(1)$ time complexity enables this approach to be highly scalable.

2) *Distributed Storage*: It was shown in [19], [21] that the number of components in Equation (9) is bounded by $H \leq N^2 - 2N + 2$. Therefore, the memory requirement is $O(N^3 \log N)$ for storing up to $O(N^2)$ permutation matrices with $N \log N$ bits per matrix. However, with $N = 1024$, about 10 Gbits of storage would be required, which is obviously not feasible. But with a distributed scheduling approach, each input linecard is only responsible for selecting a packet from its own VOQs to serve at each time slot. Therefore, we only need to store the i^{th} row of each permutation matrix π_h at input linecard i. As a result, the memory requirement for each input linecard is reduced to $O(N^2 \log N)$ for storing up to $O(N^2)$ rows with $\log N$ bits per row, $1/N^{th}$ of the storage required. With $N = 1024$, about 10 Mbits of storage is required, which is achievable in SRAM.

V. Frame-Based Decomposition

A. Decomposition via Edge Coloring

In cases where a fixed frame size is applicable, the decomposition problem can be greatly simplified. Specifically, let $\Lambda = (\lambda_{ik})$ be an admissible arrival traffic rate matrix where each entry λ_{ik} satisfies $0 \leq \lambda_{ik} \leq 1$. Suppose there is a frame size T, where T is an integer, such that $\Psi = \Lambda T$ contains only integers or entries that can be rounded off into integers with acceptable roundoff error. Then the Birkhoff-von Neumann decomposition procedure and online scheduling algorithm outlined in Section IV-A are unnecessarily complex.

Alternatively, it is well-known that the Slepian-Duguid [34] algorithm can perform the decomposition of $\Psi = \Lambda T$ into T permutation matrices in $O(N^2 T)$ time. For instance, if the chosen frame size T is a constant multiple of N, then the decomposition complexity is bounded by $O(N^3)$. However, the decomposition complexity can be reduced by formulating the decomposition as an edge coloring problem.

Theorem 6: Let $\Lambda = (\lambda_{ik})$ be an $N \times N$ admissible arrival traffic rate matrix, and let T be an integer frame size such that $\Psi = \Lambda T$ contains only integers. Then Λ can be decomposed in $O(NT \log T)$ time into T permutation matrices such that

$$\Lambda \leq \sum_{h=1}^{T} \pi_h \tag{13}$$

Proof: The matrix $\Psi = \Lambda T$ can be transformed into a bipartite graph with N nodes $u_1 \dots u_N$ and $v_1 \dots v_N$ on each side, respectively, with ψ_{ik} edges from u_i to v_k for all i and k. We then solve an edge coloring problem, which can be solved in $O(E \log D)$ time [35], [36]. Since the number of edges is $E = NT$ and the maximum degree $D = T$, the edge coloring problem can be solved in $O(NT \log T)$ time with T colors. For each color, a permutation matrix can be induced by the edges with that color. ∎

The advantages of the proposed frame-based decomposition approach over Birkhoff-von Neumann decomposition in the cases where a fixed frame size is applicable are as follows:

1) The offline decomposition complexity is $O(NT \log T)$. For instance, if the chosen frame size T is a constant multiple of N, then the decomposition complexity is bounded by $O(N^2 \log N)$. This is a much lower complexity than the $O(N^{4.5})$ complexity required by the Birkhoff-von Neumann decomposition.
2) No need to first make the traffic rate matrix doubly stochastic via an $O(N^3)$ von Neumann conversion.
3) No need to use PGPS scheduling since the T permutation matrices can be uniformly rotated in constant time, for instance using [37].
4) The online memory requirement per input linecard reduces to $O(T \log N)$, or $O(N \log N)$ when T is a constant multiple of N. This is a much lower memory complexity than the Birkhoff-von Neumann decomposition approach that generates up to $O(N^2)$ permutation matrices.

B. Delay of Frame-Based IMS

In this section, we compare the end-to-end delay of the frame-based approach using the IMS architecture with the frame-based scheme described in [12], since both schemes require a fixed frame size. In the frame-based scheme described in [12], packets arrive in frames, and packets that arrive within a frame of T time slots must satisfy the specified rate matrix $\Lambda = (\lambda_{ik})$. Specifically, let $\psi_{ik} = \lambda_{ik} T$. Then no more than ψ_{ik} packets from input i to output k can arrive within a frame. Assuming these assumptions hold, it was shown in [12] that the maximum end-to-end delay for all arrivals is bounded by $2NT$ or $\Theta(NT)$. If T is a constant multiple of N, then the bound becomes $\Theta(N^2)$.

Theorem 7: Following the same assumption that packets arrive in frames, with size T, and no more than ψ_{ik} packets from input i to output k arrive within a frame, we can guarantee that the maximum end-to-end delay for all arrivals is bounded by $T + 2N - 1$, i.e. $\Theta(T + N)$.

Proof: Following the same assumption that packets arrive in frames and no more than ψ_{ik} packets from input i to output k arrive within a frame, we are guaranteed that a packet will be scheduled for transfer no more than T time slots later. From Equation (4), once a packet has been scheduled for transfer, it will depart at its final output $2N - 1$ additional time slots later. Therefore, the maximum end-to-end delay is bounded by $T + 2N - 1$ or $\Theta(T + N)$. ∎

If T is a constant multiple of N, then the bound becomes $\Theta(N)$, which is significantly better than $\Theta(N^2)$ required by the frame-based scheme described in [12].

VI. Example and Results

In this section, we first present an illustrative example showing how the IMS architecture operates. We then present simulation results to verify our theoretical results and observations in the previous sections.

A. An Example

We illustrate in this section how an IMS emulates an IQ switch, and more specifically a Birkhoff-von Neumann

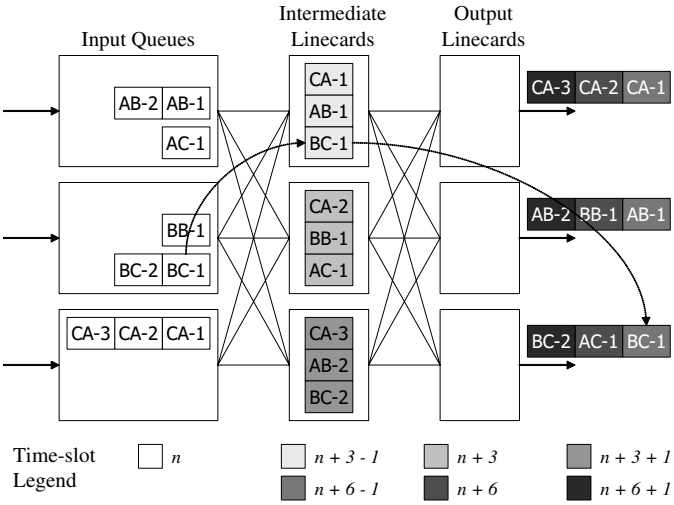

Fig. 4: Snapshots of switch in different time slots.

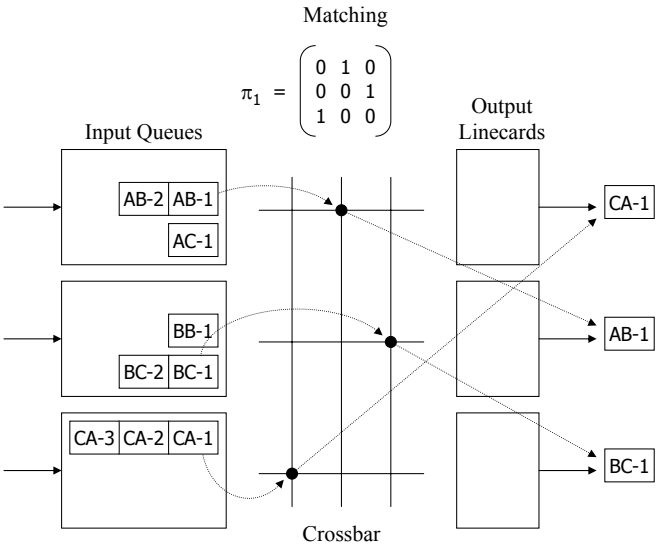

Fig. 5: IQ switch implementation of a given matching.

switch. Specifically, we illustrate by means of an example how matchings generated via online scheduling of a Birkhoff-von Neumann decomposition are executed. Consider the following 3×3 example.

$$\Lambda = \begin{pmatrix} 0 & 0.75 & 0.25 \\ 0 & 0.25 & 0.75 \\ 1.0 & 0 & 0 \end{pmatrix}$$

A possible decomposition is

$$\Lambda = 0.75 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 0.25 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

A possible online schedule may be the following matching sequence in time

$$\pi_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \pi_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\pi_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \pi_4 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\vdots \qquad \vdots$$

where π_1, π_3, π_4 are schedules of the first component, π_2 is a schedule of the second component, and so on.

The flow of packets for this matching sequence is illustrated in Figure 4. In the figure, the $N = 3$ ports are labelled A, B, and C. Each packet is labelled with its input source, output destination, and sequence number – e.g. packet BC-1 originates from input B, is destined for output C, and has sequence number 1. Figure 4 depicts *snapshots* of the switch at different points in time. Suppose that at time slot n, the packets in *white* are queued at the input stage. By the end of time slot $n+3-1$, packets CA-1, AB-1, and BC-1 completely arrive at middle linecard A, as shown in the *next shade of gray* in the figure. Other packets *in-flight* are not explicitly animated. By the end of the next time slot, $n+3$, packets CA-2, BB-1, and AC-1 completely arrive at middle linecard B. This is shown with the *next darker shade of gray*. Still other packets are *in-flight*, and so on. By the end of time slot $n+6-1$, packets CA-1, AB-1, and BC-1 completely arrive at their respective output linecards where they depart. Consider in particular the packet path for BC-1: it leaves input linecard B at time n, arrives at middle linecard A at time $n+3-1$, and departs from output linecard C at time $n+6-1$. Therefore, this example illustrates how packets are scheduled in a simple way that exactly emulates the corresponding Birkhoff-von Neumann switch.

In comparison, the flow of packets in a traditional IQ switch is illustrated in Figure 5. The figure shows how at each time-slot, the crossbar needs to be reconfigured to match inputs and outputs according to the current matching. The crossbar then services the input queues accordingly. In particular, the transfer of packets shown in Figure 5 for the IQ switch is emulated by the IMS shown in Figure 4 by transferring the same packets to the first intermediate linecard shown in Figure 4.

B. Simulation Results

In this section, we demonstrate the performance of the IMS architecture by means of simulations. The results are shown in Figures 6, 7, 8, and 9.

In particular, since the main advantage of the IMS architecture is its ability to provide service guarantees, we compare results using the IMS architecture with another load-balanced switch architecture proposed in [12] that can provide service guarantees. Specifically, a frame-based load-balanced switch architecture was proposed in [12] that can provide service guarantees with $O(1)$ online complexity. The results for the IMS architecture and the frame-based load-balanced switch architecture are labeled as “IMS" and “Frame-LBvN" in the figures, respectively.

In addition, we have also included simulation results for the originally-proposed load-balanced switch [1], which does not provide packet ordering guarantees, and the two modified

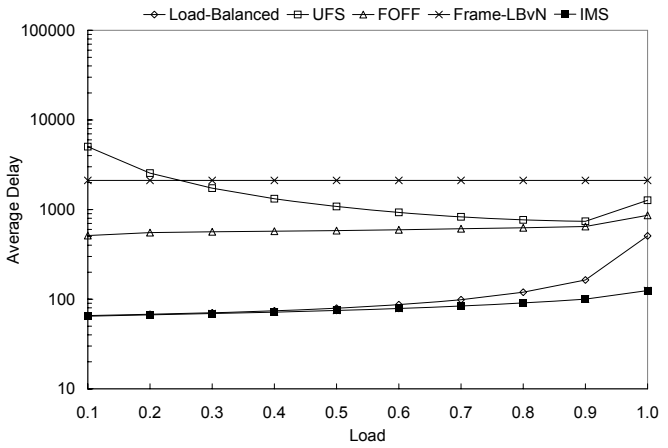

Fig. 6: Average delay for uniform traffic under a frame period of $T = 2N = 64$. Switch size is $N = 32$.

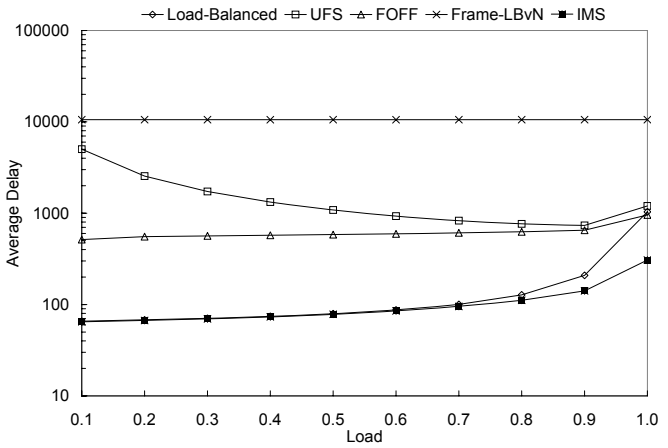

Fig. 7: Average delay for uniform traffic under a frame period of $T = 10N = 320$. Switch size is $N = 32$.

load-balanced switch architectures proposed in [6], which do provide packet ordering guarantees. The first method is called uniform frame spreading (UFS), and the second method is called full-ordered frame first (FOFF).

For comparison, we consider simulations using a uniform traffic model where packets arriving to each input have a uniform distribution of output destinations. That is, the probability that a packet arrives at input i with output destination k is uniformly ρ/N, where ρ is the input load. For the Frame-LBvN switch architecture to work [12], it assumes that an admissible rate matrix $\Lambda = (\lambda_{ik})$ and a *fixed* frame period T are both given. In this case, Λ is simply the uniform rate matrix. It also assumes that the number of packets arriving at input i with output k is *bounded* by $\psi_{ik} = \lambda_{ik}T$ during any frame period of T consecutive time slots to guarantee packet ordering. To satisfy this assumption, we used a modified random arrival process as follows: whenever $\psi_{ik} = \lambda_{ik}T$ packets have already been generated between input i and output k during a frame period, then no more packets between input i and output k will be further generated in the random arrival process for the remaining time slots in the frame. This modified random arrival process is used in all simulations to provide a common basis for comparisons. Also, we consider a common switch size of $N = 32$ in all simulations.

For the first set of results shown in Figure 6, we used a frame period of $T = 2N = 64$. Several observations can be made in this first set of results.

- First, as discussed in the paper, the delay of the Frame-LBvN switch is bounded by $\Theta(NT)$, whereas the delay for the IMS architecture is bounded by $\Theta(T + N)$. This explains why the average delays for the IMS architecture are significantly lower than the average delays for the Frame-LBvN switch. The long delays required by the Frame-LBvN switch are due in part to the aggregation of packets into frames used by the switch.
- Second, to achieve packet ordering guarantees, the UFS and FOFF methods are also based on the aggregation of packets into frames, and they are known to have $O(N^3)$ and $O(N^2)$ delay bounds, respectively. For the case when the traffic is known, the IMS architecture can achieve significantly lower average delays while ensuring packet ordering, as shown in Figure 6.
- Finally, although the original load-balanced switch is able to achieve low average delays, it does not guarantee packet ordering. This can be detrimental to Internet traffic since the widely used TCP transport protocol falsely regards out-of-order packets as indications of congestion and packet loss.

The same trends can be seen for different switch sizes.

For the second set of results, we used a larger frame period of $T = 10N = 320$ time slots. These results are shown in Figure 7. As can be seen with the results, the delay for the IMS architecture increased with the larger frame size under high loads. However, the most substantial increase in delay can be seen with the Frame-LBvN method as the increased in frame size is amplified by a factor of N. Therefore, the advantage of the IMS architecture over the Frame-LBvN switch is much more pronounced when larger frame sizes are required. Again, the same trends can be seen for different switch sizes.

Finally, in order to test the performance of IMS with non-uniform traffic, we consider simulations using a *diagonal* non-uniform traffic model, as illustrated in Figures 8 and 9. Specifically, we consider a diagonal traffic pattern in which

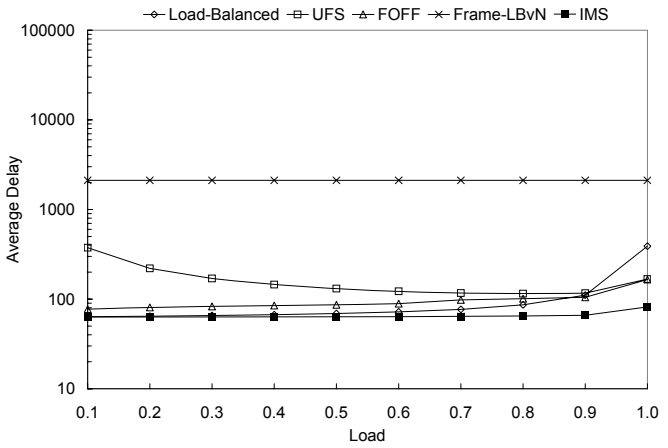

Fig. 8: Average delay for diagonal traffic under a frame period of $T = 2N = 64$. Switch size is $N = 32$.

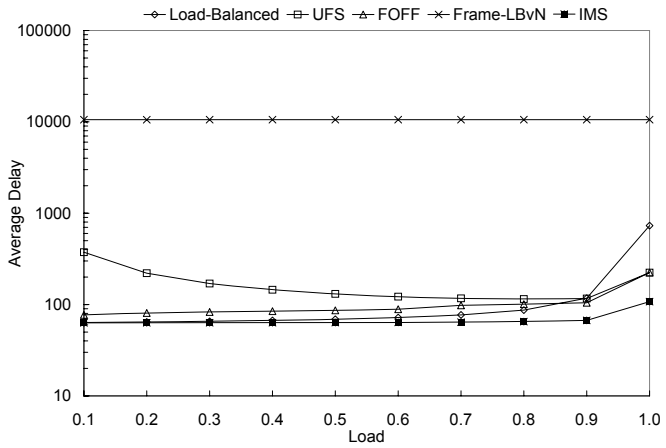

Fig. 9: Average delay for diagonal traffic under a frame period of $T = 10N = 320$. Switch size is $N = 32$.

input i sends three quarters of its traffic to itself and one quarter of its traffic to its right neighbor. That is, for all i, $\lambda_{ii} = 3\rho/4$ and $\lambda_{ik} = \rho/4$, where $k = (i+1) \mod N$. For all other i and k, $\lambda_{ik} = 0$. We again use the modified random arrival process described above to ensure that the number of packets arriving at input i with output k is bounded $\psi_{ik} = \lambda_{ik}T$ during any frame period of T consecutive time slots. In these results, we again see a similar trend as the results for the uniform traffic case. IMS still has the lowest delays even though the traffic matrix is sparse, which makes it easier for UFS and FOFF to accumulate full frames, as there are fewer flows that spread traffic. Further, we again see that the delay of the Frame-LBvN switch is bounded by $\Theta(NT)$. The delays of all the other switching solutions under comparison have even lower delays than in the uniform traffic case.

VII. Conclusions

In this paper, we proposed the interleaved matching switch (IMS) as a scalable two-stage switch architecture that guarantees packet ordering. From a scalability perspective, IMS uses the same two stages of fixed uniform meshes as in current load-balanced switch architectures. These fixed uniform meshes do not require arbitrary per-packet switch configurations and are amenable to scalable implementation in optics. We showed that IMS can emulate any IQ or CIOQ switches by interleaving the corresponding matchings across the intermediate input linecards of a load-balanced switch. With this emulation result, many of the throughput and service guarantees provided in the literature for IQ and CIOQ switches directly extend to the IMS architecture. Using the emulation result, for the case of any known admissible traffic, we showed that IMS can provide at $O(1)$ online complexity both service and stability guarantees by emulating a Birkhoff-von Neumann switch. In the case where a fixed frame period is applicable, we presented an efficient decomposition algorithm based on edge coloring. Finally, our work on the IMS architecture connects a large, well-developed, and still progressing body of work on scheduling algorithms to load-balanced switch architectures. We believe this connection opens the possibility for a great deal more research in this direction.

References

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

[2] C. S. Chang, D. S. Lee, and C. M. Lien, "Load balanced Birkhoff-von Neumann switches, Part II: multi-stage buffering," *Computer Commun.*, vol. 25, pp. 623-634, 2002.

[3] I. Keslassy, and N. McKeown, "Maintaining packet order in two-stage switches," *IEEE Infocom '02*, NY, June 2002.

[4] I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," *ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[5] I. Keslassy, S. T. Chuang, and N. McKeown, "A load-balanced switch with an arbitrary number of linecards," *IEEE Infocom '04*, Hong Kong, 2004.

[6] I. Keslassy, "The Load-Balanced Router," *Ph.D. Thesis*, Stanford University, 2004.

[7] I. Keslassy, C.-S. Chang, N. McKeown and D.-S. Lee, "Optimal load-balancing," *IEEE Infocom '05*, Miami, FL, Mar. 2005.

[8] B. Lin, and I. Keslassy, "A scalable switch for service guarantees," *IEEE Hot Interconnects XIII*, Stanford, CA, Aug. 2005.

[9] B. Lin, and I. Keslassy, "The concurrent matching switch architecture," *IEEE Infocom '06*, Barcelona, Spain, Apr. 2006.

[10] B. Lin, and I. Keslassy, "Frame-aggregated concurrent matching switch," *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Orlando, FL, December 3-4, 2007.

[11] J. J. Jaramillo, F. Milan, and R. Srikant, "Padded frames: A novel algorithm for stable scheduling in load-balanced switches," to appear in *IEEE/ACM Trans. Networking*.

[12] C.-S. Chang, D.-S. Lee and C.-Y. Yue, "Providing guaranteed rate services in the load balanced Birkhoff-von Neumann switches," *IEEE/ACM Trans. Networking*, vol. 14, no. 3, pp. 644-656, 2006.

[13] Y. Shen, S. Jiang, S.-S. Panwar, and H.-J. Chao, "Byte-Focal: A practical load-balanced switch," *IEEE HPSR*, Hong Kong, May 2005.

[14] 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," *IEEE Globecom*, 2005.

[15] C. S. Chang, D. S. Lee, and Y. J. Shih, "Mailbox switch: a scalable two-stage switch architecture for conflict resolution of ordered packets," *IEEE Trans. Commun.*, vol. 56, no. 1, pp. 136-149, 2008.

[16] T. T. Lee, and C. H. Lam, "Path switching-a quasi-static routing scheme for large scale ATM packet switches," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 914-924, 1997.

[17] T. Weller, and B. Hajek, "Scheduling nonuniform traffic in a packet switching system with small propagation delay," IEEE/ACM Trans. Networking, vol. 5, no. 6, pp. 813-823, 1997.

[18] A. Hung, G. Kesidis, and N. McKeown, "ATM input-buffered switches with guaranteed-rate property," *IEEE ISCC*, Athens, pp. 331-335, 1998.

[19] C. S. Chang, W. J. Chen, and H. Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," *IEEE IWQoS'99*, pp. 79-86, London, UK, 1999.

[20] C. S. Chang, W. J. Chen, and H. Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *IEEE Infocom '00*, pp. 1614-1623, Tel Aviv, Israel, 2000.

[21] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univers. Nac. Tucuman Rev. Ser. A,* vol. 5, pp. 147-151, 1946.

[22] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions Theory Games*, vol. 2, pp. 5-12, Princeton University Press, Princeton, NJ, 1953.

[23] M. Shreedhar, and G. Varghese, "Efficient fair queueing using deficit round robin," *ACM SIGCOMM*, pp. 231-242, Sept. 1995.

[24] S. Ramabhadran, and J. Pasquale, "Stratified round robin: a low complexity packet scheduler with bandwidth fairness and bounded delay," *ACM SIGCOMM*, pp. 239-250, Karlsruhe, Germany, 2003.

[25] R. Bhagwan, and B. Lin, "Fast and scalable priority queue architecture for high-speed network switches," *IEEE Infocom '00*, Tel Aviv, vol. 2, pp. 538-547, Mar. 2000.

[26] H. Wang, and B. Lin, "Pipelined van Emde Boas tree: Algorithms, analysis, and applications," *IEEE Infocom '07*, Anchorage, AK, May 2007.

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

[28] N. McKeown, V. Anantharan, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Infocom '96*, vol. 1, pp. 296-302, San Francisco, CA, Mar. 1996.

[29] N. McKeown, "Scheduling algorithms for input-queued cell switches," *Ph.D. Thesis*, University of California, Berkeley, 1995.
[30] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1030-1039, 1999.
[31] H. Attiya, D. Hay, and I. Keslassy, "Packet-mode emulation of output-queued switches," *ACM Symp. Parallelism Algorithms Architectures (SPAA '06)*, Cambridge, MA, August 2006.
[32] A. K. Parekh, and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: The single-node case," *IEEE/ACM Trans. Networking*, vol. 1, pp. 334-357, 1993.
[33] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithms," *ACM SIGCOMM*, pp. 1-12, Austin, TX, 1989.
[34] J. Hui, *Switching and traffic theory for integrated broadband networks*, Kluwer Academic Publishers, Boston, 1990.
[35] R. Cole, K. Ost, and S. Schirra, "Edge-coloring bipartite multigraphs in $O(E \log D)$ time," *Combinatorica*, vol. 21, no. 1, pp. 5-12, 2001.
[36] T. Takabatake, "Another simple algorithm for edge-coloring bipartite graphs," *IEICE Trans. Fundamentals Electron., Commun. Computer Sciences,* vol. E88-A, no. 5, pp. 1303-1304, 2005.
[37] T. T. Lee, "The Kraft's inequality of scheduling for packet-switched Clos networks," *IEEE Infocom mini-conference*, Phoenix, AZ, 2008.

Bill Lin holds a BS, a MS, and a Ph.D. degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley. He is currently on the faculty of Electrical and Computer Engineering at the University of California, San Diego, where he is actively involved with the Center for Wireless Communications (CWC), the Center for Networked Systems (CNS), and the California Institute for Telecommunications and Information Technology (CAL-IT^2) in industry-sponsored research efforts. Prior to joining the faculty at UCSD, he was the head of the System Control and Communications Group at IMEC, Belgium. IMEC is the largest independent microelectronics and information technology research center in Europe. It is funded by European funding agencies in joint projects with major European telecom and semiconductor companies. His research has led to over 100 journal and conference publications. He has received a number of publication awards, including the 1995 IEEE Transactions on VLSI Systems Best Paper award, a Best Paper award at the 1987 ACM/IEEE Design Automation Conference, Distinguished Paper citations at the 1989 IFIP VLSI Conference and the 1990 IEEE International Conference on Computer-Aided Design, a Best Paper nomination at the 1994 ACM/IEEE Design Automation Conference, and a Best Paper nomination at the 1998 Conference on Design Automation and Test in Europe. He also holds 2 awarded patents.

Isaac Keslassy (M'02) received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2000 and 2004, respectively.

He is currently a faculty member in the electrical engineering department of the Technion, Haifa, Israel. His recent research interests include the design and analysis of high-performance routers and on-chip networks. He is the recipient of the Yigal Alon Fellowship and of the ATS-WD Career Development Chair.