A Scalable Switch for Service Guarantees

Bill Lin* Isaac Keslassy**

*University of California, San Diego, La Jolla, CA 92093–0407. Email: billlin@ece.ucsd.edu
**Technion – Israel Institute of Technology, Haifa 32000, Israel. Email: isaac@ee.technion.ac.il

***Abstract*— Operators need routers to provide service guarantees such as guaranteed flow rates and fairness among flows, so as to support real-time traffic and traffic engineering. 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 admissible traffic. In cases where fixed frame sizes are applicable, we also present an efficient 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.

I. Introduction

A. Background

There has been recent interest in a class of switch architectures called *load-balanced routers* [2]–[6]. 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. 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 arrival. Thus, there is no need for arbitrary per-packet dynamic switch configuration, which is extremely difficult to achieve at high-speeds. The use of fixed configuration switches are particularly amendable 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 [2]–[5] 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.

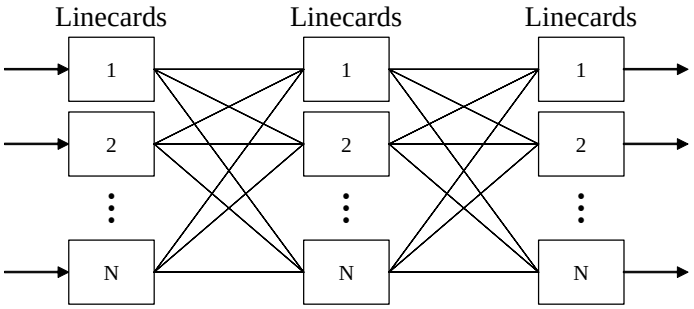

Fig. 1. Generic load-balanced switch architecture.

The objective of this paper is to investigate an approach for providing service guarantees that retains the key aspects of a load-balanced switch architecture.

In [6], two 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 [6], 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 F. The end-to-end delay of this approach is $\Theta(NF)$, 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.

B. Contributions of the Paper

The paper has three major contributions. 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. Consequently, many of the throughput and service guarantees provided in the literature for IQ and CIOQ switches directly extend to load-balanced switches.

Second, we show that our proposed IMS architecture can practically provide service guarantees by *emulating* a Birkhoff-von Neumann input-queued switch [1]. In the IMS architecture,

the switch interconnection patterns are fixed and independent of packet arrival. This is in contrast to crossbar-based Birkhoff-von Neumann switches that require arbitrary per-packet dynamic switch configurations, which are difficult to scale to high capacities and line rates. In addition to using scalable fixed configuration switches, an IMS only requires a fully-distributed online fair queueing mechanism at each input linecard with $O(\log N)$ complexity, based only on local state information. The online scheduling complexity can be reduced to $O(1)$ time by using either a round-robin based scheduler [13], [14] or a pipelined-sorting mechanism [12] with a timestamp-based scheduler. Since the IMS architecture guarantees that packet ordering is maintained throughout the switch, it does not require packet resequencing at the output or the associated resequencing delays. IMS can provide uniform service guarantees for all non-uniform admissible traffic where the traffic profile is known.

Third, 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, and the online scheduling step simply requires constant time operation. In contrast to the frame-based scheme presented in [6] that has an $\Theta(NF)$ end-to-end delay bound, we show that our approach has a significantly lower $O(F + N)$ end-to-end delay bound. In particular, if F is a constant integer multiple of N, then the frame-based scheme presented in [6] incurs an $\Theta(N^2)$ end-to-end delay bound, whereas our approach only incurs an $\Theta(N)$ end-to-end delay bound.

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 an IQ switch, which in turn can emulate a Birkhhoff-von Neumann switch to provide service guarantees, as developed in Section IV. Further, in cases where a fixed frame size is applicable, Section V describes a frame-based scheme that has low end-to-end delay. Section VI shows how the IMS architecture can be extended to emulate any CIOQ switch as well. Section VII provides an illustrative example. Finally, Section VIII briefly outlines how the architecture can be optimized to improve delay bounds.

II. The Interleaved Matching Switch

A. Overview of the Architecture

This section describes the IMS architecture. Note that throughout this paper, we assume that packets have a fixed length and time is slotted.

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 [2]–[5]. 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. The final stage consists of N output linecards where packets depart.

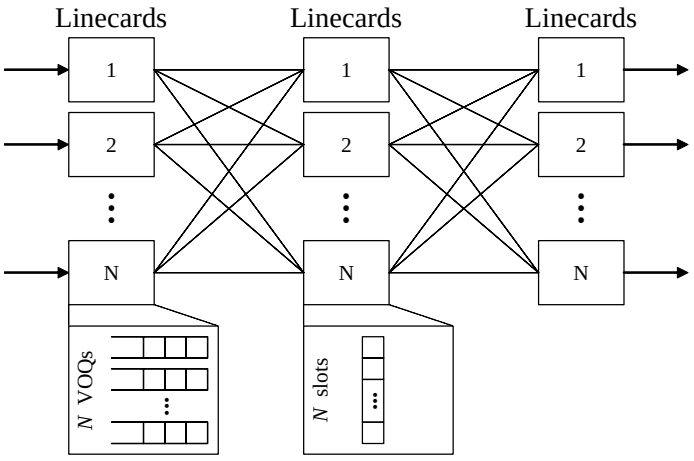

Fig. 2. The interleaved matching switch architecture.

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 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], [5], 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 n through a line of rate R will be completely transferred by the end of time slot n, 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 $n + N - 1$.

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 n. The packets are immediately queued at their appropriate VOQs, depending on their final output destination.

At the start of every time slot n, after packet arrivals, each input linecard i selects a VOQ Z_{ik} to service, 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

$$j = ((n-1) \mod N) + 1. \quad (1)$$

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

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

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

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

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

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

and the packet departs immediately from the router.

The above process operates continuously. Specifically, at time slot $n = 1$, 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 $n = 2$, 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 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.

III. IMS Can Emulate Any IQ Switch

In this section, we prove that the IMS architecture can emulate any IQ switch using the same matching algorithm m. There are two main reasons behind our desire to emulate IQ switches. The first reason is a 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 [1], which is known to provide service guarantees with only a low-complexity online scheduler that can be fully distributed.

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 $\pi^m(n)$ represent the matching of all inputs to outputs under the matching algorithm m at time slot n, and let $k = \pi^m(i, n)$ denote the matching from input i to output k under m at time slot n. Then $\pi^m(n)$ is said to be a conflict-free matching if and only if $i_1 \neq i_2 \Rightarrow \pi^m(i_1, n) \neq \pi^m(i_2, n)$.

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 $MT(c)$ be the time slot that it gets matched for transfer from input linecard i. Then $MT(c)$ 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 $DT(c)$ be the time slot that it completely arrives at output linecard k where it departs. Then $DT(c)$ 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 VII.

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 recurrence, 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 $MT^X(c) = MT^Y(c)$.

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,

$$DT^Y(c) = MT^Y(c). \quad (5)$$

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

$$DT^X(c) = MT^X(c) + 2N - 1. \quad (6)$$

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

$$DT^X(c) - DT^Y(c) = 2N - 1. \quad (7)$$

■

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.

IV. Providing Service Guarantees

A. Background on Birkhoff-von Neumann Decomposition

In [1], an approach based on Birkhoff-von Neumman 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 $N \times N$ 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* if

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

and it is said to be *doubly stochastic* if

$$\sum_{i=1}^{N} \lambda_{ik} = 1, \forall k \qquad \sum_{k=1}^{N} \lambda_{ik} = 1, \forall i$$

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

$$\Lambda \leq \sum_{h=1}^{H} \phi_h \pi_h \quad (8)$$

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 [1] consists of the following:

1) It first uses an $O(N^3)$ von Neumman algorithm [7] to convert an admissible traffic rate matrix into a doubly stochastic matrix.
2) It then uses an $O(N^{4.5})$ Birkhoff decomposition algorithm [8] to find the decomposition shown in Equation 8. Both steps 1 and 2 are performed *offline*.
3) It finally uses the Packetized Generalized Process Sharing (PGPS) algorithm in [10], [11] 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.

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

It follows directly from Section III that an IMS can provide the same throughput and service guarantees as a Birkhoff-von Neumann input-queued switch.

Theorem 2: An IMS can emulate a Birkhoff-von Neumann IQ switch.

Proof: The result derives directly from Theorem 1. ■

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 3: If the arrival traffic is a stationary and ergodic stochastic process with the 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], [2]. It extends directly to the IMS architecture using the emulation proved in Theorem 2. ■

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

Theorem 4: 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 packets served from a continually backlogged flow $\mathcal{F}_{ik}$ 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

$$C^X_{ik}(t_2) = C^Y_{ik}(t_2 - (2N-1)), \quad (9)$$

$$C^X_{ik}(t_1) = C^Y_{ik}(t_1 - (2N-1)). \quad (10)$$

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 [1], 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 Neumman switch for any $t_1 \leq t_2$ is bounded by

$$\sum_{h \in E_{ik}} \phi_h(t_2 - t_1) - \sigma_{ik} \leq C_{ik}(t_2) - C_{ik}(t_1) \quad (11)$$

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

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 8. Therefore, it follows that an IMS that emulates a Birkhoff-von Neumann switch has 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 Neumman 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. If sequential hardware is used, the PGPS scheduling step requires $O(\log N)$ complexity. When $O(\log N)$ parallel hardware can be afforded, then it has been shown in [12] that this online scheduling step can be reduced to $O(1)$ time with pipelining. Alternatively, another practical solution is to use an $O(1)$ complexity round-robin based scheduler [13], [14] to approximate PGPS scheduling. The combination of the IMS architecture and fully distributed online scheduling with $O(\log N)$ or $O(1)$ time complexity enables this approach to be highly scalable.
2) **Distributed Storage**: It was shown in [1], [8] that the number of components in Equation 8 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 service 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 Scheme

A. Low Complexity Decomposition

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 is within the interval $0 \leq \lambda_{ik} \leq 1$. Suppose there is a frame size F, where F is an integer, such that $\Psi = F \cdot \Lambda$ 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 [15] algorithm can perform the decomposition of $\Psi = F \cdot \Lambda$ into F permutation matrices in $O(N^2 F)$ time. If the chosen frame size F is an integer multiple of N, then the decomposition complexity is bounded by $O(N^3)$. However, the decomposition complexity can be further improved by formulating decomposition as an edge-coloring problem.

Theorem 5 (Frame-Based Decomposition): Let $\Lambda = (\lambda_{ik})$ be an $N \times N$ admissible arrival traffic rate matrix, and let F be an integer frame size such that $\Psi = F \cdot \Lambda$ contains only integers. Then Λ can be decomposed in $O(NF \log F)$ time into F permutation matrices such that

$$\Lambda \leq \sum_{h=1}^{F} \pi_h \tag{12}$$

Proof: The matrix $\Psi = F \cdot \Lambda$ can be transformed into a bipartite graph with N nodes $u_1 \ldots u_N$ and $v_1 \ldots v_N$ 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 [9]. Since the number of edges is $E = NF$ and the maximum degree $D = F$, the edge coloring problem can be solved in $O(NF \log F)$ time with F colors. For each color, a permutation matrix can be induced by the edges with that color. ■

If the chosen frame size F is an integer multiple of N, then the decomposition complexity is bounded by $O(N^2 \log N)$. The advantages of this approach over Birkhoff-von Neumann decomposition in the cases where a fixed frame size is applicable are as follows:

1) Much lower complexity than an $O(N^{4.5})$ Birkhoff 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 F permutation matrices can be uniformly rotated in constant time.
4) The online memory requirement per input linecard reduces to $O(F \log N)$, or $O(N \log N)$ if F is an integer multiple of N.

B. End-to-End Delay

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 [6] since both schemes require a fixed frame size. In the frame-based scheme described in [6], packets arrive in frames, and packets that arrive within a frame of F time slots must satisfy the specified rate matrix $\Lambda = (\lambda_{ik})$. Specifically, let $\psi_{ik} = \lambda_{ik} F$. 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 [6] that the maximum end-to-end delay for all arrivals is bounded by $2NF$ or $\Theta(NF)$. If F is an integer multiple of N, then the bound becomes $\Theta(N^2)$.

Theorem 6 (End-to-End Delay): Following the same assumption that packets arrive in frames, with size F, 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 $F + 2N - 1$ or $O(F + N)$.

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 F 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 $F + 2N - 1$ or $O(F+N)$. ■

If F is an integer 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 [6].

VI. IMS Can Emulate Any CIOQ Switch

For completeness, we extend in this section the results in Section III 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 $MT(c) = n$.

At every time slot, an intermediate input linecard j is chosen in the same way as described in Section III (i.e. *cf.* 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, which 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 $OQAT(c) = t$.

Note that this $t = n + 2N - 1$ is the same as in Equation 4 defined in Section III. 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 7 (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 recurrence, both input stages have the same arrival and departure processes, and same state. Therefore, for every packet c scheduled in Y, match time $MT_X(c) = MT_Y(c)$, and departure time at Y is $OQAT_Y(c) = MT_Y(c)$. For X, since there is a constant $2N-1$ delay through the switch from match time to arrival at destinated output, $OQAT_X(c) = MT_X(c) + 2N - 1$. Therefore, the difference in arrival time to the destined output queue is also constant:

$$OQAT_X(c) - OQAT_Y(c) = 2N - 1. \tag{13}$$

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. ■

VII. An Example

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

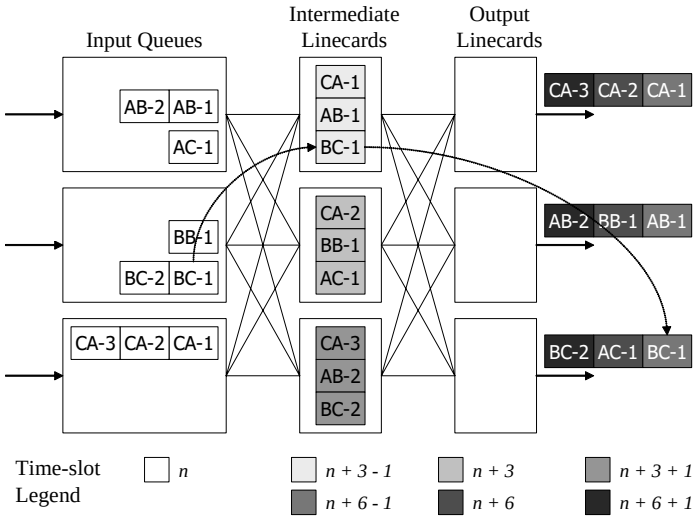

Fig. 3. Snapshots of switch in different time slots.

$$\pi(1) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \pi(2) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\pi(3) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \pi(4) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\vdots \qquad \vdots$$

where $\pi(1), \pi(3), \pi(4)$ are schedules of the first component, $\pi(2)$ is a schedule of the second component, and so on.

The flow of packets for this matching sequence is illustrated in Figure 3. 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 3 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.

VIII. Delay Optimizations

In this section, we informally outline how propagation delays can be improved. As noted in [5], 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). This approach can also be applied to the IMS architecture. In the case of IQ emulation, each logical input would be connected to each logical intermediate input by a channel at rate $2R/N$ instead of R/N via the first logical mesh, and each logical intermediate input would also be connected to each logical output by a channel at rate $2R/N$ via the second logical mesh. The channels in the first logical mesh can be *time-multiplexed* with the channels in the second logical mesh. In this case, we can reduce the fixed delay in Equation 4 by a factor of 2, and results from Sections III, IV, and V extend accordingly.

In the case of CIOQ emulation with speedup S, if the two meshes are not combined, each input is connected to each intermediate input by a channel at rate SR/N via the first mesh, and each intermediate input is connected to each output by a channel also at rate SR/N via the second mesh. Packets arriving at an intermediate input linecard can be sent to the destinated outputs immediately without waiting for all S packets to arrive. Therefore, we can reduce delay by a factor of S when using a speedup S. Similarly, if the two meshes are combined, then each logical input would be connected to each logical intermediate input by a channel at rate $2SR/N$ via the first logical mesh, and each logical intermediate input would be connected to each logical output by a channel at rate $2SR/N$ via the second logical mesh. Therefore, we can reduce delay by a factor of $2S$. The analysis from Section VI extends accordingly.

IX. Acknowledgments

The authors would like to acknowledge the support of the ATS-WD Career Development Chair.

References

[1] C. S. Chang, W. J. Chen, 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, 1999.

[2] C. S. Chang, D. S. Lee, Y. S. Jou, "Load balanced Birkhoff-von Neumann switches, Part I: one-stage buffering," Computer Communications, 2002.

[3] C. S. Chang, D. S. Lee, C. M. Lien, "Load balanced Birkhoff-von Neumann switches, Part II: multi-stage buffering," Computer Communications, vol. 25, pp. 623-634, 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, 2003.

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

[6] C. S. Chang, D. S. Lee, C. Y. Yue, "Providing Guaranteed Rate Services in the Load Balanced Birkhoff-von Neumann Switches," INFOCOM'03.

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

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

[9] R. Cole, K. Ost, S. Schirra, "Edge-coloring bipartite multigraphs in $O(E \log D)$ time," Combinatorica 21 (1) (2001) 5-12.

[10] A. K. Parekh, R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: The single-node case," IEEE/ACM Transactions on Networking, vol. 1, pp. 334-357, 1993.

[11] A. Demers, S. Keshav, S. Shenkar, "Analysis and simulation of a fair queueing algorithms," ACM SIGCOMM, pp. 1-12, Austin, TX, 1989.

[12] R. Bhagwan, B. Lin, "Fast and scalable priority queue architecture for high-speed network switches," IEEE INFOCOM, Tel Aviv, 2000.

[13] M. Shreedhar, G. Varghese, "Efficient fair queueing using deficit round robin," ACM SIGCOMM, pp. 231–242, Sept. 1995.

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

[15] J. Hui. "Switching and traffic theory for integrated broadband networks," Kluwer Academic Publishers, Boston, 1990.