Frame-Aggregated Concurrent Matching Switch

Bill Lin
University of California, San Diego,
La Jolla, CA 92093–0407, USA
billlin@ece.ucsd.edu

Isaac Keslassy*
Technion – Israel Institute of Technology,
Haifa 32000, Israel
isaac@ee.technion.ac.il

ABSTRACT

Network operators need high-capacity router architectures that can offer scalability, provide throughput and performance guarantees, and maintain packet ordering. However, previous router architectures based on centralized crossbar-based architectures cannot scale to fast line rates and high port counts. Recently, a new scalable router architecture called the Concurrent Matching Switch (CMS) [5] was introduced that offers scalability by utilizing a fully distributed architecture based on two identical stages of fixed configuration meshes. It has been shown that fixed configuration meshes can be scaled to very fast line rates and high port counts via optical implementations. It has also been shown that the CMS architecture can achieve 100% throughput and packet ordering with only sequential hardware and $O(1)$ amortized time complexity operations at each linecard. However, no delay performance guarantees have been shown for CMS.

In this paper, we demonstrate a general delay performance guarantee for CMS. Based on this guarantee, we propose a novel frame-based CMS architecture that can achieve a performance guarantee of $O(N \log N)$ average packet delay, where N is the number of switch ports, while retaining scalability, throughput guarantees, packet ordering, and $O(1)$ time complexity. This architecture improves upon the best previously-known average delay bound of $O(N^2)$ given these switch properties. We further introduce several alternative frame-based CMS architectures.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Routers*

General Terms

Algorithms, Performance, Design.

Keywords

Concurrent Matching Switch, Frame Scheduling, Packet Switching, 100% Throughput, Load-Balanced Router.

*Supported in part by an Alon Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ANCS'07, December 3–4, 2007, Orlando, Florida, USA.
Copyright 2007 ACM 978-1-59593-945-6/07/0012 ...$5.00.

1. INTRODUCTION

1.1 Motivation

Network operators need high capacity router architectures that can offer scalability, provide throughput and performance guarantees, and maintain packet ordering. However, previous crossbar-based router architectures with centralized scheduling and arbitrary per-packet dynamic switch configurations cannot scale to fast line rates and high port counts. Recently, a new scalable router architecture called the Concurrent Matching Switch (CMS) was introduced that can meet the requirements of scalability, throughput guarantees, low average packet delays, and packet ordering. This router architecture belongs to a class of scalable switch architectures that are based on two identical stages of fixed configuration uniform meshes for routing packets, including various versions of the load-balanced routers [1, 2, 3, 4].

Figure 1 shows a diagram of the CMS architecture, which consists of 3 stages of linecards that are sandwiched between two identical stages of fixed configuration uniform meshes. These fixed configuration meshes are the same ones used in load-balanced routers, which have been shown to be scalable to very high capacities and line rates [3]. The first mesh connects the first stage of input linecards to the center stage of intermediate input linecards, and the second mesh connects the center stage of intermediate input linecards to the final stage of output linecards. These fixed configuration meshes implement deterministic interconnection patterns that are independent of packet arrivals. Thus, there is no need for arbitrary per-packet dynamic switching configurations, which can be extremely difficult to achieve at high-speeds. Fixed configuration meshes are particularly amenable to scalable implementations with optics, as exemplified by the 100 Tb/s reference design described in [3]. That reference design was based on a fixed hierarchical mesh of optical channels that interconnects $N = 640$ linecards, each operating at a rate of $R = 160$ Gb/s.

To ensure 100% throughput guarantee and packet ordering, the CMS architecture uses a fully distributed

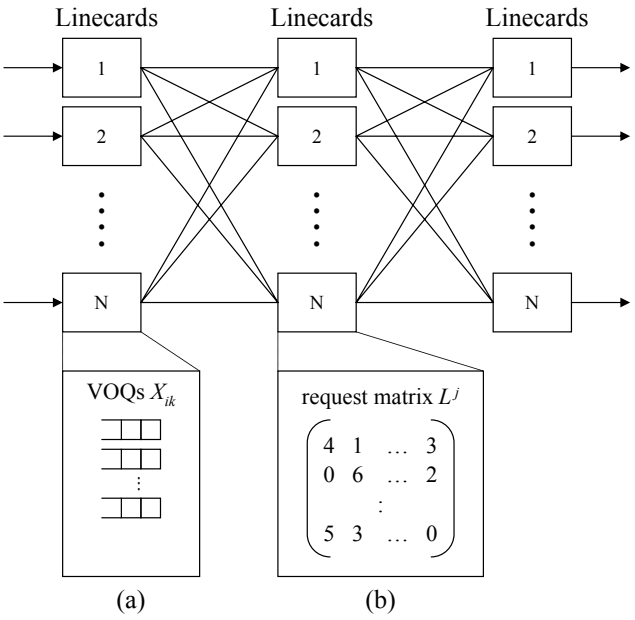


Figure 1: The Concurrent Matching Switch architecture. (a) Input linecard. (b) Intermediate input linecard.

contention-resolution mechanism. In particular, packets are buffered at the input linecards in virtual output queues. Instead of load-balancing *packets*, a CMS load-balances *request tokens* to each intermediate input linecard where each intermediate input linecard *concurrently* solves a *matching* problem based only on its local token count. By each intermediate input linecard solving a local matching problem in parallel, each intermediate input linecard independently selects a virtual output queue from each input linecard to service such that the packets selected can traverse the two fixed configuration meshes in parallel without conflicts. Packets from selected virtual output queues in turn depart in order from the input linecards, through the intermediate input linecards, and finally through the output linecards.

Each intermediate input linecard has N time slots to perform each matching, so the complexity of existing matching algorithms can be *amortized by a factor of* N. The exchanges of tokens and packets occur over the two fixed uniform meshes without the need for arbitrary dynamic switch configurations, and all queueing and decision-making functions are performed locally at each linecard using only local state information.

In [5], it was shown that 100% throughput can be guaranteed in a CMS architecture by using *any* stable matching algorithm at the intermediate input linecards. In particular, it was shown that a class of stable randomized matching algorithms [6, 7] with $O(N)$ operations can be amortized to $O(1)$ time complexity using only sequential hardware and local state information at each linecard.

Using these stable randomized matching algorithms, low average packet delays can be empirically achieved. However, no theoretical delay bounds have ever been shown for these randomized scheduling algorithms in the CMS architecture. In this paper, we will demonstrate such delay bounds in the CMS architecture. To do so, we will use the frame-based scheduling algorithm introduced in [8] for crossbar switches, which works by scheduling packets in frames of T consecutive time slots. We propose to extend this frame-based scheduling method to the CMS architecture.

1.2 Summary and Contributions of the Paper

In this paper, we first show that the fair-frame scheduling algorithm can be improved to $O(N \log \log N)$ amortized complexity per time slot by formulating the matrix decomposition problem as an *edge coloring* problem [9]. This result is applicable to the conventional crossbar scheduling problem as well.

We next show that the fair-frame scheduling algorithm, and our improved edge coloring version, can both be directly applied to the CMS architecture, since the CMS architecture can use *any* stable scheduling algorithm at the intermediate input linecards. Because the matching problem is load-balanced across N parallel schedulers, the *amortized* time complexities of the fair-frame scheduling algorithm and our improved version are $O(\sqrt{N} \log N)$ and $O(\log \log N)$, respectively. We also show that the average packet delay of the CMS architecture with these scheduling algorithms is $O(N \log N)$ under any admissible Bernoulli i.i.d. traffic. More broadly, we present a general delay bound for the CMS architecture under different scheduling algorithms.

Finally, we show that the matrix decomposition step is unnecessary if we modify the CMS architecture to operate in a *frame-aggregated* manner. Rather than returning *grant* tokens one at a time, we propose to modify the CMS architecture so that up to T grant tokens are returned together from each intermediate linecard to each input linecard, and each input linecard sends in return up to T packets to the corresponding intermediate linecard, from where the packets depart through the second switching stage to their final outputs. By eliminating the need for matrix decomposition, the time complexity of our *Frame-Aggregated CMS* architecture reduces to just $O(1)$ at each linecard, thus achieving 100% throughput, packet ordering, $O(1)$ complexity, as well as $O(N \log N)$ delay bounds under any admissible Bernoulli i.i.d. traffic. Note that to the best of our knowledge, given packet ordering and constant time complexity, no architecture could previously guarantee an average delay bound under $O(N^2)$ (which was provided in [3]).

1.3 Organization of the Paper

The rest of the paper is organized as follows. We first provide background reminders on the basic CMS architecture and the fair-frame switch scheduling algorithm in Section 2. Then, we present general delay bounds in the CMS architecture in Section 3. In Section 4, we describe an asymptotically more efficient version of the fair-frame scheduling algorithm by means of edge coloring, and we show how both the fair-frame scheduling algorithm and the new edge-coloring frame decomposition algorithm can be applied to the basic CMS architecture to achieve $O(N \log N)$ delay. In Section 5, we describe a modified frame-based CMS architecture, called a Frame-Aggregated CMS, that avoids the need for matrix decomposition, which enables it to achieve $O(1)$ time complexity. In Section 6, we present simulation results for our various frame-based schemes. Finally, we conclude the paper in Section 7.

2. BACKGROUND

In this section, we provide several background reminders needed to understand the paper. We first start with a high-level summary of the CMS architecture [5], which forms the basis of the architecture used in this paper. Then, we summarize the fair-frame scheduling algorithm presented in [8] in the context of crossbar switching, which can achieve $O(\log N)$ average packet delay with a scheduling complexity of $O(N^{1.5} \log N)$. In this paper, we will use algorithms based on the fair-frame scheduling algorithm in order to improve the CMS architecture by providing better delay guarantees and lower complexity.

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

2.1 The Basic CMS Architecture

The CMS architecture consists of three linecard stages that are interconnected by two fixed uniform meshes, exactly like the load-balanced switch architecture described in [1, 2, 3]. The CMS architecture is depicted in Figure 1. Intuitively, it is based on the notion that each of the N intermediate input linecards sees $1/N^{th}$ of the traffic and behaves as an autonomous switch scheduler for this part of the traffic. More precisely, a high-level overview of the switch operation is as follows:

1. In the basic load-balanced switch architecture proposed in [1], incoming packets are *uniformly load-balanced* across the N intermediate input linecards at the center stage where packets are buffered. Instead, in the CMS architecture, incoming packets are mainly buffered in virtual output queues at each input linecard. Specifically, each input linecard i maintains N virtual output queues, X_{i1}, ..., X_{iN}, one per output destination, as shown in Figure 1(a). Incoming packets to input i destined for output k are buffered at their virtual output queue X_{ik} immediately upon arrival.

2. Instead of spreading packets across the center stage, a key idea in the CMS architecture is to first spread *request tokens* to the intermediate input linecards at the center stage instead of actual packets. Each request token acts as a *placeholder*. The actual packets are transferred later, based on *matching* decisions that are made by the intermediate input linecards. Specifically, each input linecard is periodically connected to a given intermediate input linecard every N time-slots. For each incoming packet to input i destined for output k, a request token $r(i,k)$ is immediately generated and sent to the intermediate input linecard that is currently connected to the input. In other words, the input linecard load-balances request tokens among intermediate linecards in a cyclical way that is influenced by the arrival time of each packet.

3. When a request token $r(i,k)$ is received at an intermediate input linecard j, the corresponding token counter at intermediate input j gets incremented. Specifically, each intermediate input linecard j maintains a *request matrix* $L^j = (l^j_{ik})$, as shown in Figure 1(b). Each entry l^j_{ik} corresponds to a counter that maintains the number of request tokens at intermediate linecard j from input i to output k that has not yet been granted.

4. Each intermediate input linecard then concurrently, and independently, solves a *matching* problem based on its own request matrix that it maintains locally. It does not need any global state information or any virtual token count information from any other intermediate input linecard. Any bipartite matching algorithm may be used with the CMS architecture to perform this matching step, leveraging the well-developed body of work in this area. Each intermediate input linecard has N time slots to perform each matching step, and thus the algorithmic complexity of the matching algorithm used may be *amortized* by a factor of N.

5. Based on the result of the matching step, each intermediate input j sends at most one *grant token* $g(i,j,k)$ to each input i over the first mesh. The grant token indicates that the request matrix entry l^j_{ik} is positive and that the corresponding virtual output queue X_{ik} has been matched. In other words, the grant token indicates that there was a demand and that the demand has been answered. In addition, each matrix entry $l^j_{i,k}$ for which a grant token is generated is decremented.

6. In response to the grant token $g(i,j,k)$ received, each input i then sends the packet at the head of the corresponding virtual output queue X_{ik} to intermediate input j over the first mesh. Note that this packet is not necessarily the one that sent the request token which triggered the grant token $g(i,j,k)$, even though it is guaranteed that to each grant token corresponds at least one packet waiting to depart; in fact, in order to maintain packet ordering, tokens are formally associated to queues, not to specific packets. Next, the (up to) N packets sent by the inputs are then temporarily stored in a set of coordination packet buffers at each intermediate input on their path to the outputs. As soon as the packets are fully received, each of the the intermediate input linecards j forwards them over the second mesh to the output linecards.

7. Finally, packets are received at output linecard k where they depart immediately from the router.

Using the above operation, the CMS architecture *guarantees packet ordering*. Moreover, it is proved to be *strongly stable* as long as a strongly stable matching algorithm is used for Step 4 above. In particular, for any admissible Bernoulli i.i.d. arrival traffic, CMS guarantees that the number of packets queued in the switch is not expected to grow to infinity. The proof relies on the interesting fact that the token traffic received by any intermediate input during N time-slots has the same distribution as the packet traffic received by the router during a single time-slot. Incidentally, note that the periodic token allocation scheme described in Step 2 is vulnerable to adversarial traffic patterns. If arriving traffic is not Bernoulli i.i.d., a random non-periodic token allocation will be preferred.

As described in [5], the two stages of uniform meshes used in the CMS architecture can be replaced by a single mesh, and the 3 *logical* linecards (input, intermediate input, and

output) can be combined into a single physical linecard. Using a single combined mesh, each pair of combined linecards are interconnected by two fixed rate data channels and two fixed rate control channels. Each data channel operates at a fixed rate of R/N, where R is the external line rate to each switch port. The two control channels are used to transfer the request and grant tokens. Each control channel only needs to operate at a rate of $\epsilon R/N$, where ϵ is the ratio of the token size to the fixed packet size, typically only about 1-2%. Combining all four channels together, each pair of combined linecards are interconnected with a combined bandwidth of $2(1+\epsilon)R/N$.

2.2 Fair-Frame Scheduling Algorithm

The main idea of the fair-frame scheduling algorithm proposed in [8] is the following. Given some random (Poisson or Bernoulli i.i.d.) packet arrivals, there exists an integer frame size of T consecutive time slots such that the probability of oversubscription at any output is negligible. It was shown in [8] that, for a specified demand load upper-bound ρ and a switch size N, logarithmic average delay can be achieved if we choose the minimum frame size

$$T = \left\lceil \frac{\log(2N/\delta)}{\log(1/\gamma)} \right\rceil, \quad (1)$$

where $\gamma = \rho e^{1-\rho}$, and the probability of overflow δ satisfies

$$\delta\,(1/\rho + N + N\rho T) < 1. \quad (2)$$

It was also shown that $\delta = O(1/N^2)$ can be chosen such that T remains $O(\log N)$.

Let $L(f)$ be the arrival matrix under random input for frame f of T consecutive time slots. The fair-frame scheduling algorithm works by scheduling $L(f)$ on a frame-by-frame basis as follows:

1. In the first frame, the initial permutation matrices are chosen at random.
2. In the $(f+1)^{th}$ frame, the maximum row and column sum T^* is computed for the previously arrived frame $L(f)$.
3. If $T^* \leq T$, then $L(f)$ is augmented with null packets to form $\tilde{L}(f)$ so that all row and column sums are T^*. Otherwise, any arriving packet that exceeds T packets to an output is stripped off from $L(f)$ to overflow queues to form $\tilde{L}(f)$.
4. $\tilde{L}(f)$ is scheduled during frame $(f+1)$ by means of maximum size matching at each time slot, which is guaranteed to clear $\tilde{L}(f)$ in T^* time slots.
5. If $T^* < T$, uniform random scheduling is performed on the remaining slots to serve the overflow queues. Repeat from Step 2.

The main complexity is in Step 4, where the complexity of maximum matching is $O(M\sqrt{N})$ [10], with M being the number of non-zero entries in $\tilde{L}(f)$, which is $O(N \log N)$ since T is $O(\log N)$. Therefore, the complexity of Step 4 is $O(N^{1.5} \log N)$. Two optimizations to the above algorithm were outlined in [8]. First, in Step 3, instead of augmenting $\tilde{L}(f)$ with null packets, it can be augmented with packets from the overflow queues. Second, in Step 5, if $T^* < T$, and the overflow queues are empty, then dynamic frame sizing can be employed by starting immediately on the next frame.

3. DELAY BOUNDS

In [5], it was shown that any scheduling algorithm can be used with the CMS architecture, and that a CMS is strongly stable if the corresponding scheduling algorithm used is strongly stable. This section expands the theoretical results in [5] by analyzing the delay of a CMS.

Let's first formally define *strong stability*. Assume that the packet arrival process to each input is Bernoulli i.i.d., and that the probability that a packet arrives to input i for output k at any time-slot is provided by a traffic matrix $\Lambda = (\lambda_{ik})$. Further, assume that the arrival matrix is strictly doubly sub-stochastic (admissible), i.e., there exists some demand load $\rho < 1$ such that for all i, k,

$$\sum_{i=1}^{N} \lambda_{ik} \leq \rho, \sum_{k=1}^{N} \lambda_{ik} \leq \rho. \quad (3)$$

We can now introduce the definition of strong stability.

Definition 1 (Strong Stability). *A switch is said to be* strongly stable *if under the Bernoulli i.i.d. admissible packet arrival process defined above, the number of packets queued in the switch is not expected to grow to infinity, i.e.,*

$$\limsup_{n\to\infty} E\left[\sum_{i,k} X_{ik}(n)\right] < \infty. \quad (4)$$

A matching algorithm for a crossbar switch that can achieve strong stability is also said to be strongly stable.

The following theorem establishes the average delay of a CMS with a strongly stable matching algorithm.

Theorem 1. *Given an admissible Bernoulli i.i.d. arrival process, let σ be a strongly stable matching algorithm with average packet delay (waiting time) of W_σ in a single switch. Then a CMS using σ is also strongly stable, with an average delay of $O(NW_\sigma)$.*

Proof. The strong stability was proved in [5], so let's now prove the delay part (note that the delay result implies the strong stability as well, and thus is a stronger result). First, we will define a new internal time reference for the intermediate input linecards. At each intermediate input j, tokens can only be received and granted (respectively packets can only arrive and depart) *every N time-slots*. Therefore, at each intermediate input, we will cut time into frames of N time-slots. As detailed in the proof of strong stability [5], the intermediate input linecard operates at every frame in CMS with algorithm σ as it would have at every time-slot in a single switch with the same algorithm σ. Therefore, under the same arrival pattern of request tokens in CMS (respectively of packets in a single switch), it takes the same number of frames for grant requests to exit intermediate input linecards (respectively the same number of time-slots for packets to exit the single switch) under the same algorithm σ. In addition, in both cases, the traffic arrival matrix is indeed the same, and therefore all traffic arrival characteristics of this Bernoulli i.i.d. traffic are the same as well. Therefore, if the average packet waiting time is defined as W_σ time-slots in the single switch, it will be

exactly W_σ frames in the intermediate input linecard, corresponding to $N \cdot W_\sigma$ time-slots. Further, the additional fixed propagation times in the CMS architecture are all in $O(N)$ (request tokens take a bounded time to arrive to the intermediate input linecards, grant tokens to arrive to the inputs, and finally packets to travel through the meshes). Finally, $W_\sigma \geq 1$ (assuming the scheduling result comes at least one time-slot after the packet arrivals), therefore the total delay is $N \cdot W_\sigma + O(N) = O(N \cdot W_\sigma)$. □

Incidentally, it is worth noting that under admissible Bernoulli i.i.d. traffic, given some fixed load, the average delay of output-queued switches is known to be constant, independent of N [8]. Further, using a speedup of two, matching algorithms that can emulate output-queued switches have been shown [11]. Therefore, a corollary of Theorem 1 is that using a speedup of two, CMS can achieve an $O(N)$ average packet delay.

4. FRAME-BASED SCHEDULING

In this section, we first show that the CMS architecture can implement the fair-frame scheduling algorithm to achieve $O(N \log N)$ average delay with $O(\sqrt{N} \log N)$ amortized time complexity. Then, we improve on the time complexity by proposing an alternative frame decomposition formulation based on edge coloring that can achieve the same logarithmic delay bound, but with a lower $O(\log \log N)$ complexity.

4.1 Frame Scheduling for CMS

As proved in [5], any scheduling algorithm can be used with the CMS architecture — and for any strongly stable scheduling algorithm, the corresponding CMS is also strongly stable. Further, it has been shown in [8] that the fair-frame scheduling algorithm is strongly stable. Therefore, a CMS based on the fair-frame scheduling algorithm is strongly stable as well. We next analyze the average packet delay of the CMS architecture using fair-frame scheduling.

THEOREM 2. *CMS achieves $O(N \log N)$ average delay using fair-frame scheduling.*

PROOF. Follows from Theorem 1. Note that due to the $\Theta(N \log N)$ frame length, this is also a lower bound for non-trivial traffic patterns. □

THEOREM 3. *The amortized time complexity of CMS using fair-frame scheduling at each linecard is $O(\sqrt{N} \log N)$.*

PROOF. It was shown in [5] that the complexity of any scheduling algorithm is *amortized* by a factor of N. Given that the complexity of fair-frame scheduling is $O(N^{1.5} \log N)$, it follows that the amortized complexity of CMS using fair-frame scheduling is $O(\sqrt{N} \log N)$ at each linecard. □

4.2 Frame Decomposition via Edge Coloring

In this section, our objective is to provide the same delay guarantees as with the above fair-frame scheduling algorithm, but with a smaller complexity. The key idea is that we replace the on-the-fly maximum matches done in Step 4 of the fair-frame scheduling (section 2.2) by using *edge coloring*. We will first apply the results to the typical single-crossbar switch case, and then expand them to the CMS architecture.

Our objective is to perform a *frame decomposition* of $\tilde{L}(f)$ into a sequence of T^* permutation matrices. However, instead of scheduling $\tilde{L}(f)$ during frame $(f+1)$ by means of maximum matching, the actual sequencing of the permutation matrices derived for $\tilde{L}(f)$ is not carried out until during frame $(f+2)$. This way, the frame decomposition of $\tilde{L}(f)$ by means of edge coloring can be performed during frame $(f+1)$, thus amortizing its time complexity over several time-slot.

THEOREM 4. *The complexity of decomposing $\tilde{L}(f)$ in a single-crossbar switch is $O(N \log \log N)$.*

PROOF. A bipartite graph can be constructed from $\tilde{L}(f)$. The complexity of edge coloring is $O(E \log D)$ [9] where E is the number of edges and D is the maximum degree. Since the maximum row or column sum in $\tilde{L}(f)$ is bounded by T, E is bounded by NT and D is bounded by T. Therefore, the edge coloring of $\tilde{L}(f)$ can be performed in $O(NT \log T)$. Since we have T time slots to do the decomposition, then the amortized complexity reduces to $O(N \log T)$. It follows that the complexity of decomposing $\tilde{L}(f)$ is $O(N \log \log N)$ since is T is $O(\log N)$. □

The $O(N \log \log N)$ complexity improves upon the $O(N^{1.5} \log N)$ complexity at the expense of one frame delay, but the average delay remains logarithmic. We can now state that in a typical single-crossbar switch, the edge coloring frame decomposition will achieve the same asymptotic average delay as the fair-frame decomposition algorithm, with a smaller complexity.

THEOREM 5. *The edge coloring frame decomposition algorithm can achieve $O(\log N)$ average delay in a single-crossbar switch.*

PROOF. Given that $\tilde{L}(f)$ is augmented with null packets such that all row and column sums are T^*, it is well known that edge coloring produces a sequence of exactly T^* maximum matchings. Conversely, any sequence of T^* maximum matchings of $\tilde{L}(f)$ is a valid edge coloring of $\tilde{L}(f)$. Let S_{EC} be a switch that uses edge coloring frame decomposition. Let S_{FF} be the same switch using the fair-frame scheduling algorithm instead, and let $\hat{S}_{FF}$ correspond to the switch S_{FF} with its inputs delayed by a fixed delay of T. Since the scheduling of $\tilde{L}(f)$ occurs during frame $(f+2)$ with edge coloring frame decomposition instead of during frame $(f+1)$ with on-the-fly maximum matching, it is easy to see that the output behavior of S_{EC} is identical to that of $\hat{S}_{FF}$. Given that S_{FF} has been proven to achieve $O(\log N)$ average delay, it follows that $\hat{S}_{FF}$ also achieves $O(\log N)$ average delay since the fixed delay on the inputs is also $T = O(\log N)$. Therefore, S_{EC} achieves $O(\log N)$ average delay. □

It is easy to see that the two optimizations proposed in [8] can also be similarly applied with our edge coloring frame decomposition algorithm, namely $\tilde{L}(f)$ can be augmented with packets from the overflow queues, and, after the decomposition of a frame, we can start decomposing immediately the next completed frame via edge coloring. Moreover, with edge coloring, there is no need to augment $\tilde{L}(f)$ with null packets.

We can now apply the edge coloring results to the CMS architecture. We will show that CMS using edge coloring frame decomposition achieves the same $O(N \log N)$ average delay, but a smaller amortized time complexity of $O(\log \log N)$ instead of $O(\sqrt{N} \log N)$.

THEOREM 6. *CMS achieves an $O(N \log N)$ average delay bound using edge coloring frame decomposition.*

PROOF. Follows from Theorem 1. □

THEOREM 7. *The amortized time complexity of CMS using edge coloring frame decomposition at each linecard is $O(\log \log N)$.*

PROOF. Follows from Theorem 4. □

5. FRAME-AGGREGATED CMS

In Section 4, we showed that the CMS architecture can achieve $O(N \log N)$ average delay with $O(\sqrt{N} \log N)$ or $O(\log \log N)$ amortized time complexity per linecard if fair-frame scheduling or the proposed edge coloring frame decomposition algorithm is employed as the scheduling algorithm, respectively. Although both fair-frame scheduling based on maximum matching and edge coloring frame decomposition have very low amortized time complexities, they nonetheless require extra logic or software at each linecard to implement these non-trivial algorithms. Our goal is to provide a simpler algorithm that will still achieve strong stability, packet ordering, and an $O(N \log N)$ average delay, but with an $O(1)$ complexity and easy practical implementation.

To achieve $O(1)$ complexity, we show in this section that the matrix decomposition step used in fair-frame scheduling can be eliminated if we modify the CMS architecture to operate in a *frame-aggregated* manner. Rather than decomposing L, and returning *grant* tokens one at a time, we propose to modify the CMS architecture so that up to T grant tokens are returned together from each intermediate linecard to each input linecard, and the input linecard sends in return up to T packets to the corresponding intermediate linecard, from where the packets depart through the second switching stage to their final outputs. By eliminating the need for matrix decomposition, the complexity of our *Frame-Aggregated CMS* (FA-CMS) architecture reduces to just $O(1)$ at each linecard, thus achieving 100% throughput, packet ordering, $O(1)$ complexity, as well as $O(N \log N)$ delay bounds. We will first present an overview of the switch operation of a FA-CMS. After, we will establish several important properties of the FA-CMS architecture.

5.1 Switch Operation

The FA-CMS architecture is depicted in Figure 2. It is very similar to the basic CMS architecture, using the same fixed rate uniform meshes, but with some differences in the implementation of the input and intermediate input linecards.

As in fair-frame scheduling, for a specified demand load upper-bound ρ and a switch size N, we can use Equation (1) to compute a minimum frame size T with the corresponding overflow probability δ that satisfies Equation (2). Also as in fair-frame scheduling, a frame f is defined to be T consecutive time slots. Here, we further define the notion of a

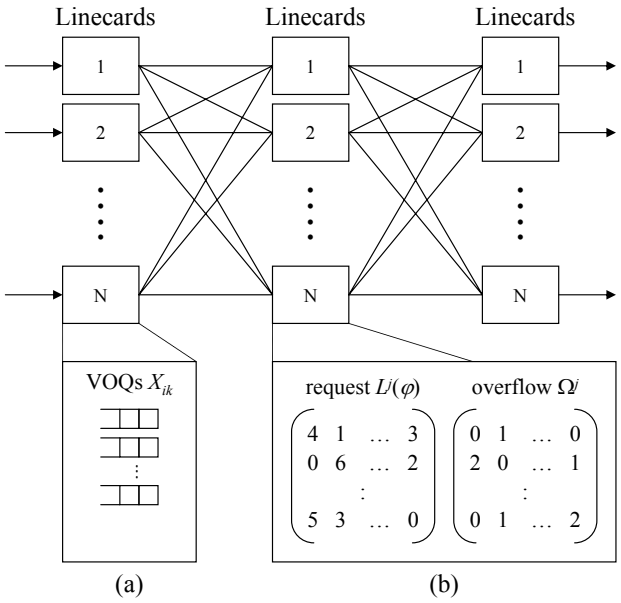


Figure 2: The Frame-Aggregated Concurrent Matching Switch architecture. (a) Input linecard. (b) Intermediate input linecard.

superframe φ consisting of N consecutive *frames* or equivalently NT consecutive *time slots*.

For the sake of simplicity, we will present the operations of the FA-CMS architecture in terms of superframes. Specifically, we will decompose the overall switch operation into four consecutive phases, each taking a superframe to complete: (1) packet arrival and transmission of request tokens, (2) transmission of grant tokens, (3) transmission of packets through the first mesh, and (4) transmission of packets through the second mesh. These phases are pipelined. They are elaborated as follows:

1. Packet arrival and transmission of request tokens:

   a. Incoming packets to input i destined for output k are buffered at their virtual output queue X_{ik} immediately upon arrival, as shown in Figure 2(a). For each arriving packet, a *request token* is generated, as in the CMS architecture.

   b. In the current superframe φ, the request tokens generated at times equal to j modulo N are sent to the j^{th} intermediate input linecard. Thus, during a superframe φ of duration NT, an intermediate input linecard j receives at most T request tokens from each input i, totaling at most NT request tokens from all N inputs.

   c. At each intermediate input linecard j, two matrices are maintained. The first is the *request matrix* $L^j(\varphi) = (l^j_{ik})$, and the second is an *overflow matrix* $\Omega^j = (\omega^j_{ik})$. For each request token $r(i,k)$ received during a superframe φ, intermediate input j increments the corresponding entry l^j_{ik} in the request matrix $L^j(\varphi)$ if the corresponding k^{th} column sum is less than T. If the corresponding k^{th} column sum has already reached T, then the corresponding entry ω^j_{ik} in

the overflow matrix is incremented instead. A running maximum row and column sum T_j^* for $L^j(\varphi)$ is updated at intermediate input j during the superframe φ each time a request matrix entry l_{ik}^j is incremented.

d. By the end of the superframe φ, all request tokens are guaranteed to have arrived at their destinated intermediate input linecards using the corresponding $\epsilon R/N$ control channels. Since in the previous step, we have already shifted all *overflow* increments to Ω^j, we are guaranteed that $T_j^* \leq T$ at the end of the superframe.

2. Transmission of grant tokens:

a. At the beginning of the $(\varphi+1)^{th}$ superframe, with $L^j(\varphi)$ and T_j^* already determined at each intermediate input linecard, each intermediate input linecard sends at most $T_j^* \leq T$ grant tokens back to each input linecard, totalling at most NT_j^* grant tokens to all N inputs. In particular, for each row i in the matrix $L^j(\varphi)$, for each non-zero entry l_{ik}^j, l_{ik}^j grant tokens $g(i,j,k)$ are sent back to input linecard i. Therefore, over a control channel with a fixed rate of $\epsilon R/N$ between each pair of intermediate input and input linecards, the transmission of at most T_j^* grant tokens to each input linecard is guaranteed to complete after NT_j^* time slots.

b. If $T_j^* < T$, then the corresponding intermediate input j will choose $V = T - T_j^*$ permutation matrices, $\pi_1, \ldots, \pi_V$, uniformly at random. For each π_v, if $\pi_v(i,k) = 1$ and $\omega_{ik}^j \neq 0$, it sends a grant token $g(i,j,k)$ to input i. Thus, with V permutation matrices, each intermediate input linecard j will send at most V additional grant tokens to each input. Therefore, again over a control channel with a fixed rate of $\epsilon R/N$ between each pair of intermediate input and input linecards, the transmission of at most V grant tokens to each input linecard is guaranteed to complete after NV time slots.

c. Therefore, in total, an intermediate input linecard j will send at most $T_j^* + V = T_j^* + (T - T_j^*) = T$ grant tokens to each input, NT grant tokens in total, and their transfers are guaranteed to complete in $NT_j^* + NV = NT$ time slots, or by the end of the $(\varphi+1)^{th}$ superframe.

3. Transmission of packets through the first mesh:

a. At the start of the $(\varphi+2)^{th}$ superframe, in the first frame, each input will first send up to T packets in FIFO order from the corresponding VOQs to the first intermediate input linecard, one per each grant token $g(i,1,k)$ received from the first intermediate input linecard. Similarly, in the j^{th} frame, each input will send up to T packets in FIFO order from the corresponding VOQs to the j^{th} intermediate input linecard, one per each grant token $g(i,j,k)$ received from intermediate input j.

b. By the end of the $(\varphi+2)^{th}$, all packets are guaranteed to arrive at the corresponding intermediate input linecards using the corresponding R/N data channels.

c. Each intermediate input linecard j will receive at most T packets from each input, totalling at most NT packets from all N inputs, and at most T packets among these NT packets will be destined to each output.

d. Since packets are sent from each input in FIFO order from the corresponding VOQs, packets from the same VOQ at the same input linecard will arrive at the destination intermediate input linecard in order.

4. Transmission of packets through the second mesh:

a. Finally, at the start of the $(\varphi+3)^{th}$ superframe, in the first frame, each intermediate input will send up to T packets that it received from the previous phase to the first output linecard in the order that they were received. Similarly, in the j^{th} frame, each intermediate input will send up to T packets to the j^{th} output linecard.

b. By the end of the $(\varphi+3)^{th}$, all packets are guaranteed to arrive at the corresponding output linecards using the corresponding R/N data channels.

c. Finally, packets received at output linecard k depart immediately from the router.

5.2 Properties

We will now establish that the FA-CMS architecture guarantees packet ordering, strong stability under random inputs, and $O(N \log N)$ average delay. To do so, we will first show that it can actually be emulated by a specific CMS architecture with fair-frame scheduling.

DEFINITION 2 (EMULATION). *A switch is said to* emulate *another switch if under identical inputs, the departure times for identical packets are identical within some bound independent of the traffic pattern.*

THEOREM 8. *CMS with fair-frame scheduling can emulate FA-CMS.*

PROOF. As defined above, in both architectures, the same request tokens are generated at the same time by the same packets for the same intermediate inputs. Now, it is possible to redefine the fair-frame scheduling in CMS to have it provide the same permutation matches as FA-CMS. (Note that all results with the fair-frame scheduling do not assume any specific maximum matching algorithm.) In the same way, it is possible to redefine it to have it select the same request tokens to mark as overhead. Thus, both architectures can eventually generate the very same grant tokens to the very same input linecards at the same times, and therefore the inputs in both architectures will eventually decide to send the same packets at the same time. Therefore, seen from the inputs, the two CMS-based architectures behave in exactly the same way, even though the algorithm is of course implemented differently in both. Finally, there is now a difference in the delays that packets need to reach the outputs and depart from the switch. Assuming the same implementation of the control channels and of the meshes, CMS with fair-frame scheduling only needs a propagation delay through the 2 meshes of $2N$, while FA-CMS has to wait for the superframe boundaries to ensure ordering, and so it needs a

total delay bounded by $2(N+N*T)$ after receiving the grant tokens. Therefore, while the difference could be large, it is independent of the specific traffic pattern and is bounded by $2NT$. Using Definition 2, CMS with fair-frame scheduling emulates FA-CMS. □

THEOREM 9. *FA-CMS guarantees packet ordering.*

PROOF. This result follows directly from the enforcement of packet ordering throughput the switch, particularly in Steps 3(a), 3(d), and 4(a). Of course, it follows also from Theorem 8 providing emulation by CMS with fair-frame scheduling, which guarantees packet ordering. □

THEOREM 10. *FA-CMS is strongly stable.*

PROOF. Using Theorem 8, we know that FA-CMS can be emulated by CMS with fair-frame scheduling; and CMS with fair-frame scheduling is strongly stable [5]. □

THEOREM 11. *FA-CMS can achieve $O(N \log N)$ average packet delay.*

PROOF. The proof of Theorem 8 shows that FA-CMS can be emulated by CMS with fair-frame scheduling, with a maximum difference in delay bounded by $2NT = O(N \log N)$. Further, CMS with fair-frame scheduling achieves $O(N \log N)$ average packet delay by Theorem 2. The result follows. □

5.3 Complexity

We now show that the above FA-CMS switch operation can be implemented with only $O(1)$ time complexity at each linecard. Therefore, FA-CMS reaches our goal of obtaining a switch architecture that achieves 100% throughput, packet ordering, $O(1)$ complexity, and guaranteed delay bounds.

THEOREM 12. *The time complexity of the FA-CMS at each linecard is $O(1)$.*

PROOF. In phase 1, each input linecard only has to queue at most one new packet at each time slot and generate at most one corresponding request token that has to be forwarded to an intermediate input linecard. Clearly this only requires constant time operation. And at each intermediate input linecard, it will receive at most one request token per time slot. For each request token received, the updates to $L^j(\varphi)$, Ω^j, and T_j^* clearly also require only constant time.

In phase 2, the main source of efficiency is the fact that *we do not need to perform matrix decomposition by either maximum matchings or edge coloring.* We only need to generate grant tokens from $L^j(\varphi)$ and Ω^j that have already been established in the previous superframe. Since each intermediate input linecard only needs to generate at most one grant token per time slot, the processing by each intermediate input or input linecard is only constant time per time slot. For Step 2(b), choosing a permutation matrix uniformly at random is clearly a constant time operation. And since $V = T - T_j^*$ is bounded by T, we only need to choose uniformly at random at most T permutation matrices, and we have NT time slots to generate them.

Finally, for phase 3 and phase 4, each input, intermediate input, or output linecard only has to receive at most one packet and transfer at most one packet per time slot, and the corresponding queueing operations take constant time. □

Note that like in the basic CMS architecture, each linecard only relies on its local state information to perform all of its decision and queueing functions. Further, the number of queues in each linecard is upper-bounded by $O(N)$, and each of these queues is in a simple FIFO mode, requiring low control complexity. In addition, the number of packet buffers stored in these queues is upper-bounded by $O(N \log N)$ per linecard, because each intermediate linecard can receive at most NT packets from all inputs in each superframe, where T is $O(\log N)$. Hence, the FA-CMS architecture is also highly scalable.

6. SIMULATIONS

In this section, we demonstrate the performance of the frame-based CMS architectures described in this paper. In our simulations, we consider independent Bernoulli i.i.d. inputs with a port loading set to $\rho = 0.7$. For each switch size N, a minimum frame size T is computed using Equation (1) with the constraint that the corresponding overflow probability δ satisfies Equation (2). The results for both uniform and non-uniform traffic are shown in Figure 3, plotting the average packet delay in time-slots as a function of the switch size N. Results shown for uniform input traffic correspond to the case where $\lambda_{ik} = \rho/N$ for all input-output pairs. For non-uniform input traffic, we consider a diagonal traffic pattern in which $\lambda_{ik} = 2\rho/3$ when $i = k$, and $\lambda_{ik} = \rho/3$ when $((i-1) \mod N) + 1 = k$.

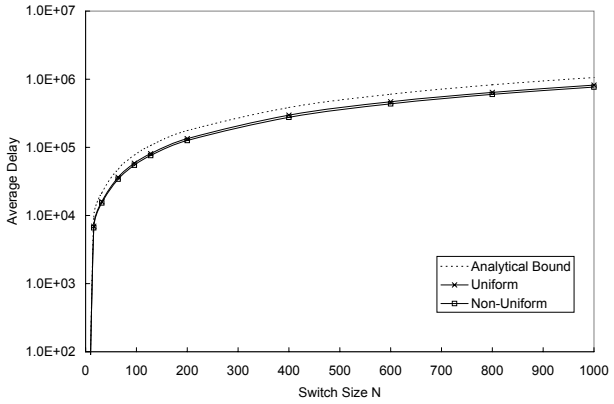
We first show in Figure 3(a) the performance of the frame-aggregated CMS architecture described in Section 5. This architecture has the advantage that it only requires $O(1)$ complexity without the need for any scheduling or decomposition algorithm. Also shown on Figure 3(a) is the plot of the analytical $O(N \log N)$ delay bound proved in Theorem 11. As can be seen in the figure, the delay results for the frame-aggregated CMS architecture fall within this bound.

We next show in Figure 3(b) the performance of the CMS architecture [5] using the fair-frame scheduling algorithm [8] and dynamic frame sizing, as described in Section 4.1. The figure illustrates that the delay for the non-uniform traffic case is slightly better than the uniform traffic case. In both cases, it can be seen that the delay results follow the same asymptotic profile as in Figure 3(a).

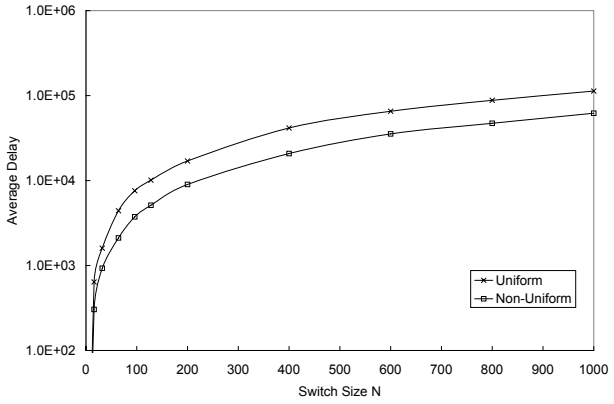
Note that the delay is longer in Figure 3(a) compared to Figure 3(b) for two reasons. First, because of the delay incurred by the aggregation of superframes between the two switch stages, as noted for instance in the proof of the emulation theorem (Theorem 8), since FA-CMS works using superframes, while CMS with the fair-frame scheduling algorithm can switch packets at the intermediate input linecards as soon as they arrive. More significantly, the fair-frame scheduling algorithm was implemented using dynamic frame sizing, yielding frame sizes that were typically smaller on average than the analytical results by a factor of up to 10.

7. CONCLUSIONS

In this paper, we presented a general delay bound for the previously proposed CMS architecture [5], and we showed that the fair-frame scheduling algorithm proposed in [8] can be combined with the CMS architecture to achieve $O(N \log N)$ average delay. We also showed an improved



(a) Frame-Aggregated CMS.



(b) CMS with Fair Frame Scheduling.

Figure 3: Simulation results with uniform and non-uniform traffic, $\rho = 0.7$.

formulation of the fair-frame scheduling algorithm by replacing the maximum matching step with an edge coloring frame decomposition step, which reduced the complexity of the algorithm from $O(N^{1.5} \log N)$ to $O(N \log \log N)$. This result is applicable to the conventional crossbar scheduling problem as well. When used in conjunction with the CMS architecture, these complexities are amortized by a factor of N to $O(\sqrt{N} \log N)$ and $O(\log \log N)$, respectively. Finally, we showed that an $O(N \log N)$ average delay can be achieved with a modified version of the basic CMS architecture without having to perform either fair-frame scheduling or edge coloring frame decomposition. Instead, we modify the basic CMS architecture to operate in a frame-aggregated manner. By eliminating the need for any matching algorithm, the time complexity of our Frame-Aggregated CMS architecture then reduces to just $O(1)$ at each linecard. In summary, using our proposed Frame-Aggregated CMS architecture, we achieve an $O(N \log N)$ average delay bound, as well as a 100% throughput guarantee, packet ordering, and $O(1)$ complexity, thus improving on the best previously-known average delay bound of $O(N^2)$ given these switch properties.

8. REFERENCES

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

[2] 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.

[3] 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.

[4] B. Lin, I. Keslassy, "A scalable switch for service guarantees," *Proceedings of the 13th IEEE Symposium on High-Performance Interconnects*, Aug 17-19, 2005.

[5] B. Lin, I. Keslassy, "The concurrent matching switch architecture," *IEEE INFOCOM*, Barcelona, Spain, April 2006.

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

[7] P. Giaccone, B. Prabhakar, D. Shah, "Randomized scheduling algorithms for input-queued switches," *IEEE Journal of Selected Areas in Communication*, vol. 21, no. 4, pp. 642-655, May 2003.

[8] M. J. Neely, E. Modiano, Y.-S. Cheng, "Logarithmic delay for N x N packet switches under the crossbar constraint," *IEEE Transactions on Networking*, vol. 15, no. 3, pp. 657-668, June 2007.

[9] R. Cole, K. Ost, S. Schirra, "Edge-coloring bipartite multigraphs in $O(E \log D)$ time," *Combinatorica*, vol. 21, no. 1, pp. 5-12, 2001.

[10] J. Hopcroft, R. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Computing*, pp. 225-231, December 1973.

[11] S. T. Chuang, A. Goel, N. McKeown, B. Prabhakar, "Matching output queueing with a combined input output queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030-1039, 1999.