The Crosspoint-Queued Switch

Yossi Kanizo
Dept. of Computer Science
Technion, Haifa, Israel
ykanizo@cs.technion.ac.il

David Hay
Dept. of Electronics
Politecnico di Torino, Turin, Italy
hay@tlc.polito.it

Isaac Keslassy
Dept. of Electrical Engineering
Technion, Haifa, Israel
isaac@ee.technion.ac.il

***Abstract*—This paper calls for rethinking packet-switch architectures by cutting all dependencies between the switch fabric and the linecards. Most single-stage packet-switch architectures rely on an instantaneous communication between the switch fabric and the linecards. Today, however, this assumption is breaking down, because effective propagation times are too high and keep increasing with the line rates.**

In this paper, we argue for a self-sufficient switch fabric by moving all the buffering from the linecards to the switch fabric. We introduce the *crosspoint-queued (CQ)* switch, a new buffered-crossbar switch architecture with large crosspoint buffers and no input queues, and show how it can be readily implemented in a single SRAM-based chip using current technology. For a crosspoint buffer size of one, we provide a closed-form throughput formula for all work-conserving schedules under uniform Bernoulli i.i.d. arrivals. Furthermore, we study the performance of the switch for larger buffer sizes and show that it nearly behaves as an ideal output-queued switch. Finally, we confirm our results using synthetic as well as trace-based simulations.

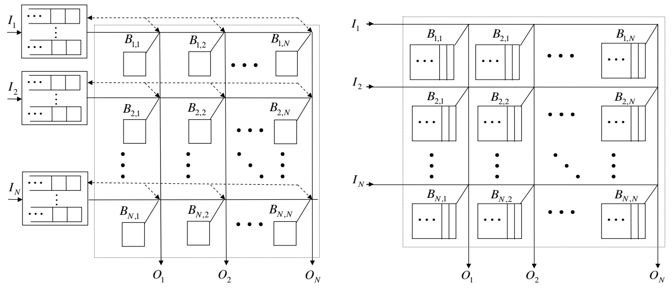
I. Introduction

A. Background

This paper is about freeing the switch fabric. We want to design a self-sufficient switching module that can switch packets without permanently relying on the current states of the input linecards, the output linecards, and the complex centralized scheduler. Ultimately, our goal is to build a *switch-on-a-chip*, i.e. a single-chip self-sufficient switching module.

Packet switch architectures are getting increasingly complex, so we should rethink whether we really need this complexity. As line rates increase to cope with demand in the Internet core and in large datacenters, the popular input-queued switch architectures prove unable to scale. Their complex centralized schedulers cannot run at line rate, requiring a costly memory speedup to hide their under-performance. As a result, they are implemented as combined-input-and-output-queued switches, with large queues in both inputs and outputs [1], [2]. Therefore, switching a packet needs complex permanent control communication between several modules spread across the switch: the input linecards, the centralized arbiter, the switch module, and the output linecards. While theory assumes that any communication between these modules is instantaneous, practice requires long and complex pipelines to mask effective delays. In short, switch complexity is a difficult problem, and higher line rates can only worsen it.

Recently, *combined-input-and-crosspoint-queued (CICQ)* switch architectures like IBM Prizma [3], [4] have generated increased interest, because they appear simpler to implement [5]–[16]. As shown in Fig. 1(a), CICQ switch architectures combine large virtual output queues (VOQs) in the input linecards with a buffered-crossbar switch fabric. This buffered crossbar typically holds a single packet buffer per crosspoint (following common practice, we consider segmented fixed-size packets). At each time slot, in a distributed manner, each input can write up to one packet into an empty crosspoint, and each output can read up to one packet from a full crosspoint. Therefore, inputs and outputs can work independently without relying on a centralized scheduler.



(a) CICQ switch (b) CQ switch

Fig. 1. Illustration of the CICQ and CQ switch architectures, with inputs I_i, outputs O_j, and crosspoint buffers B_{ij}. In the CICQ switch, the dashed lines represent the communication between the linecards and the buffers.

CICQ switches are especially appealing because they have a simple architecture that can scale to high line rates. They do not need a speedup of N, unlike output-queued and shared-memory switches [17]. They also do not need a complex centralized scheduler to match inputs and outputs, unlike most input-queued and combined-input-and-output-queued switches [1], [2]. Finally, they do not need multiple switching stages, unlike load-balanced switches and parallel packet switches [18], [19].

While CICQ architectures are attractive, they also assume the existence of an instantaneous communication channel between the linecards and the switch fabric: at each time slot, input linecards need to know whether a crosspoint is empty in order to write into it. However, since the linecard and switch fabric racks are separated to decrease the power consumption per rack [18], this assumption does not hold in practice, and the round-trip communication time cannot be neglected in front of a time-slot. For instance, the round-trip propagation time can last up to 600 ns, assuming inter-rack optical fibers with a maximum length of 60 meters and a propagation speed of $c/n = 2{\cdot}10^8$ m/s [20]. On the other hand, a time-slot lasts 2 ns, assuming 64-byte segmented packets in next-generation OC-

3072 lines (160 Gbps) with a 60% segmentation overhead [21]. Therefore, a round-trip time is clearly not negligible in front of a time-slot — in fact, it lasts some *300 time-slots*, not even counting Serializer/Deserializer (SERDES) delays, and this number keeps increasing *proportionally to the line rate* [20].

The high effective propagation delay between the input linecards and the switch fabric emphasizes the need to dissociate them and cut any instantaneous back-and-forth control communication. This is especially true for the CICQ architecture, because hiding the propagation delay would require either sizing crosspoint buffers to a round-trip time, moving the large power-consuming input queues back into the switch fabric rack, or relying on speculative algorithms without performance guarantees [9].

B. The CQ Switch: Architecture and Implementation

In this paper, we introduce the *crosspoint-queued (CQ) switch*, as illustrated in Fig. 1(b). In the CQ switch, all the buffering is moved from the linecards to the switch fabric, and therefore the input linecards do not hold input queues anymore. Furthermore, the switch fabric is a buffered crossbar switch with large crosspoint buffers.

The CQ switch relies on a self-sufficient switch fabric. As usual, each incoming packet first goes through the packet lookup-processing-segmentation pipeline at the input. Then, the incoming packet simply arrives directly at its crosspoint buffer. It is not queued before at the input and does not rely on any control information before being sent to the crosspoint buffer. If the crosspoint buffer is full, it simply drops the incoming packet. Otherwise, the packet is queued at its crosspoint buffer. Next, for each output destination, the buffered crossbar picks a non-empty buffered crosspoint (if any) and sends its head-of-line packet. Therefore, there is *no need for control communications* between the linecards and the switch fabric anymore.

Buffered crossbar switches with large crosspoint buffers were briefly considered in the ATM literature as building blocks of multi-stage switches (e.g., [22]–[25]). Unlike the CQ switch, these switches were very small (usually, 2×2 and 4×4 switches) and since they were part of a multi-stage switch, they often used traffic control mechanisms in order to avoid packet drops, and relied on large input buffers at the ingress of the multistage switch. An architecture with crosspoint buffers of size one and no input queues was also considered in [10] as an analysis tool.

The CQ switch fabric could be readily implemented on a *single chip*. While the N^2 crosspoint buffers used to be prohibitive, current crossbar switches are limited by the number of pins required to get data on and off the chip, not by the die area [8], [9], [21]. For instance, in a 128×128 CQ switch, each crosspoint buffer could hold *over* 60 *packets*, assuming an aggressive high-performance ASIC design with 64-byte segmented packets, an 18 mm $\times$ 18 mm die with 70% memory area and 30% crossbar and memory-decoder logic area [9], and an SRAM cell size of 0.4 μm^2 (ITRS numbers for 2008) [26]. Further, the crosspoint buffer size is expected to *keep growing with SRAM density*, roughly doubling every 2.5 years [26].

In addition, in order to obtain a high total throughput, switches often need to *use multiple crossbars in parallel* (by bit-slicing, time-slicing, or using multiple-stage switching). Therefore, switch designers can use these many parallel buffered crossbars to increase the total buffer size or reduce the buffer size per chip [12], [20]. For instance, keeping our example of $N = 128$ ports at 160 Gbps with a 60% segmentation overhead, we would need a total switch throughput of 34 Tbps. But each crossbar can only switch up to 640 Gbps, assuming 10 Gbps per 16 I/O pins, 5 Gbps per port, and a conservative package with 1500 pins, including 1024 I/O pins (up to 4800 pins are available today in aggressive ASIC designs using ITRS specifications) [20], [26], [27]. Therefore, independently of our switch architecture, we need some 50 parallel crossbars to get enough bandwidth. Consequently, to keep the same total buffer size, we can now adopt a very conservative design with higher die yields: for instance, by using a 10 mm $\times$ 10 mm die, with 10% memory area and an SRAM cell size of 1 μm^2.

Moreover, CQ switch buffering could be reduced even further by switching *variable-size packets* and removing packet segmentation and reassembly [9], [12], [28]–[30]. In our example, this would reduce the needed throughput and buffer size by about 35%, i.e. the packet segmentation overhead.

C. The CQ Switch: Comparative Considerations

The CQ switch presents interesting characteristics when compared to existing switch architectures. First, most practical switch architectures currently assume infinite input queueing. To the best of our knowledge, the CQ switch is *one of the first practical switch architectures without input queueing* (except for the architectures based on buffered-crossbars mentioned above). Other such architectures have too low a throughput at speedup one to be practical. In fact, the CQ switch is also one of the only switch architectures to explicitly rely on *finite buffers*, since a CQ switch with infinite buffers would simply be an output-queued switch.

Consequently, *heavy buffering disappears from input linecards*. This is one of the most significant implementation advantages of CQ switches. Today, linecard buffers take about half of their board space and a third of their power consumption [31]. They rely on massive amounts of SRAM and DRAM with fast access times, require complex scheduling algorithms to manage these SRAM and DRAM modules, and can take a significant amount of time to design [32]–[34]. Therefore, no linecard buffering also means reduced design time, complexity, and power consumption. This makes CQ switches especially suited for switch implementations in *large datacenters* (low complexity) and *networks-on-chip* (low power consumption).

In addition, the CQ switch can be seen as the theoretical continuator of the shared-memory (SM) and output-queued (OQ) switches. The SM switch shares memory dynamically between inputs and outputs, resulting in a memory speedup of N for reads and writes; the OQ switch shares memory statically between outputs and dynamically between inputs, resulting in a speedup of N for writes and 1 for reads. The CQ switch goes one step further in the static sharing: it shares memory statically between inputs and outputs, resulting in a speedup of 1 for reads and writes. Thus, in contrast with the shared-memory switch, it can be seen as a *private-memory switch*.

As a consequence of the private-memory property, the CQ switch provides a *natural flow isolation*. Unlike shared-memory switches, it inherently isolates misbehaved flows from well-behaved flows coming from different inputs and sharing the same output. Thus, it is free of the *buffer-hogging effect* [20].

Finally, recent papers argue that due to TCP mechanisms, switch buffers can be *much smaller* than previously believed, and justify the removal of the gigabits of data buffering currently present in input linecards [35]–[38]. Together with the advances in SRAM density, they make CQ switches possible today, while they were not ten years ago.

D. Performance Analysis

We introduced above the CQ switch architecture and explained how it can be implemented as a single-chip self-sufficient switching module. In this paper, we further analyze its properties with both small and large crosspoint buffers.

First, with *small crosspoint buffers* of size one, we observe that any work-conserving scheduling algorithm yields *the same performance* when the arrival traffic is Bernoulli i.i.d. and uniform. Using z-transforms, we derive a closed-form expression for the throughput and average delay of the CQ switch. This closed-form expression enables us to explore the system behavior as a function of N, and in particular to prove that the *CQ switch throughput approaches* 100% as N goes to infinity.

Later, we consider *larger crosspoint buffers* and model their performance with commonly-used scheduling algorithms: longest-queue-first (LQF), random, and exhaustive round-robin scheduling. When the buffers are larger than 1, closed-form throughput expressions are hard to obtain. Instead, we provide throughput models, and check their accuracy using simulations. For instance, we find that LQF-based CQ switches and OQ switches have close overflow probabilities, using uniform, non-uniform, as well as bursty long-range-dependent traffic.

Finally, we present simulation results in which we analyze the performance of CQ switches using synthetic inputs as well as real-life traces, and show that CQ switches typically reach a throughput close to 100% for reasonable crosspoint buffer sizes.

We now introduce the CQ switch model and notations in Section II. Then, we provide closed-form performance expressions for crosspoint buffers of size one in Section III. Later, we analyze the behavior of a CQ switch with larger crosspoint buffers and the LQF, random, and exhaustive round-robin algorithms in Sections IV, V and VI, respectively. Finally, Section VII provides simulation results using synthetic inputs as well as real-life traces.

Due to space limits, some proofs are omitted and can be found in [39].

II. The Crosspoint-Queued Switch Model

A. The Crosspoint-Queued Switch Architecture

A *crosspoint-queued (CQ)* switch is a buffered crossbar switch that has no memory but in the crosspoints of the buffered crossbar; i.e., it neither contains queues in the inputs nor in the outputs (as shown in Fig. 1(b)).

Let N be the number of switch inputs (outputs), and B the memory size of each crosspoint buffer; that is, the CQ switch has N^2 crosspoint buffers and can hold up to $B \cdot N^2$ packets. Furthermore, let B_{ij} denote the buffer that resides in the crosspoint of input i and output j.

Following common practice, we assume that time is slotted into fixed-size *time-slots* and that packets have a fixed size, i.e. segmentation and reassembly are done outside the switch [20]. Each of the time-slots is conceptually divided into two phases:

- The *arrival phase*, in which packets arrive at the switch and are stored in the crosspoint buffers. If a packet arrives at input i and is destined for output j, it is stored in buffer B_{ij} — unless B_{ij} is full, in which case the packet is dropped upon arrival.
- The *departure phase*, in which packets leave the switch.

During the departure phase, for each output, the buffered crossbar picks independently and in parallel a non-empty buffered crosspoint (if any). It then sends its head-of-line packet to the output, in order to maintain packet ordering. For simplicity, we say that *the output schedules the packet*, even though the buffered crossbar is the one to effectively select it. Each output may use *any* scheduling algorithm in order to decide which packet to serve. Scheduling algorithms considered in this paper include the LQF, random, and exhaustive round-robin algorithms.

B. Definitions and Notations

In this section, we successively define several properties of scheduling algorithms, traffic arrivals, and switch performance measures. We start with scheduling algorithms.

Definition 1: A scheduling algorithm is called *work-conserving* if each output always services a buffer whenever one of the buffers destined to it is non-empty.

Example 1: The LQF, random, and exhaustive round-robin scheduling algorithms are work-conserving.

The switch performance strongly depends on the packet arrival pattern. A common example is the Bernoulli independent and identically distributed (i.i.d.) arrival process, in which at each time-slot t, a single packet arrives at input i for output j with probability $\lambda_{ij} \geq 0$, independently of the past and of other inputs. Let the *traffic matrix* be $\Lambda = [\lambda_{ij}]$, the arrival rate at each input i be $\lambda_i^{in} = \sum_{j=1}^{N} \lambda_{ij}$, and the arrival rate at each output j be $\lambda_j^{out} = \sum_{i=1}^{N} \lambda_{ij}$. We can now define the uniform traffic process.

Definition 2: A Bernoulli i.i.d. arrival process is *uniform* with load $\rho \in [0,1]$ iff all the elements in its traffic matrix Λ are equal to $\frac{\rho}{N}$.

We now define several performance measures of a switch. Note that a switch performance depends on several parameters, including its traffic arrival pattern, its scheduling algorithm, and its size (N and B).

Definition 3: The *throughput* of a switch component is the limiting ratio of the cumulative number of packets entering it successfully by the cumulative number of packet arrivals, as time goes to infinity (whenever this limit is defined). Specifically: TP_{ij} is the *crosspoint throughput* of buffer B_{ij}, TP_{c_j} is the *column throughput* of output j, and TP is the *switch throughput*.

Definition 4: The *delay* of a switch component is the limiting average delay of packets that are traversing this component, whenever defined. Specifically: W_{ij} is the *crosspoint delay* of buffer B_{ij}, W_{c_j} is the *column delay* of output-port j, and W is the *switch delay*.

It is important to note that in CQ switches the performance of each output can often be analyzed independently, because the CQ switch architecture isolates packets destined to each output. In fact, as long as the traffic process from input i to output j is independent of other traffic processes, we will be able to study CQ switches "column-wise".

III. Crosspoints of Size One

This section investigates CQ switches with a crosspoint buffer size of one and Bernoulli i.i.d. uniform traffic of load ρ. We obtain exact closed-form expressions for the delay and the throughput of a CQ switch under these settings. Remarkably, these results hold for *any scheduling algorithm*, as long as it is work-conserving.

A. Equivalence of Work-Conserving Schedulers

Our objective is to prove that under Bernoulli i.i.d. uniform arrivals, a CQ switch with a crosspoint buffer size of one has the same performance *independently of its work-conserving scheduler*. We first demonstrate several general properties of the CQ switch before proving this result.

The next proposition points out a link between the throughput and the probability that a buffer is full in the steady state. It is true for any Bernoulli i.i.d. traffic (not necessarily uniform), and any crosspoint buffer size (not necessarily one). Its proof is essentially based on a discrete version of the PASTA property: in this case, Bernoulli arrivals see the time average of the probability that the buffer is not full.

Proposition 1: In a CQ switch with Bernoulli i.i.d. arrivals, the crosspoint throughput TP_{ij} of buffer B_{ij} is equal to the probability that a packet arriving to B_{ij} is not dropped. Equivalently, if P_{ij}^k denotes the steady-state probability that buffer B_{ij} stores k packets *before* the arrival phase, then $TP_{ij} = 1 - P_{ij}^B$.

Proof: The steady-state probability that a packet both arrives and is successfully absorbed at buffer B_{ij} in an arbitrary time slot is $\lambda_{ij} \cdot \left(1 - P_{ij}^B\right)$, since these two events are independent. By Definition 3, $TP_{ij} = \frac{\lambda_{ij} \cdot \left(1-P_{ij}^B\right)}{\lambda_{ij}}$, hence the result. ■

We use Proposition 1 to deduce the following equivalent method to evaluate the column throughput of a CQ switch using steady-state probabilities. Again, it holds for any Bernoulli i.i.d. traffic and any crosspoint buffer size.

Proposition 2: Consider a CQ switch with a work-conserving scheduler for output j. Let π_j^0 denote the probability that all the crosspoint buffers of column j are empty *after* the arrival phase. Then, the column throughput TP_{c_j} of column j is $TP_{c_j} = \frac{1-\pi_j^0}{\sum_{i=1}^N \lambda_{ij}}$.

Proof: We denote the steady-state rate of absorbed packets in all the buffers at the crosspoints of output j as γ_j, and the steady-state rate of served packets in all the buffers at the crosspoints of output j as δ_j. By Definition 3, the column throughput of output j is $TP_{c_j} = \frac{\gamma_j}{\sum_{i=1}^N \lambda_{ij}}$. In the steady state, the rate of absorbed packets is equal to the rate of served ones. Moreover, assuming a work-conserving scheduling algorithm, the rate of served packets in all the buffers at the crosspoints of output j is equal to the probability that at least one of the buffers is not empty. Therefore $\gamma_j = \delta_j = 1 - \pi_j^0$, and the result follows. ■

The next proposition characterizes the average delay through a CQ switch with $B = 1$ and uniform Bernoulli i.i.d. arrivals as a function of its throughput. It enables us to characterize the average delay of the CQ switch as well as its throughput in the following theorems.

Proposition 3: In a CQ switch with $B = 1$, uniform Bernoulli i.i.d. arrivals of load ρ, and throughput TP, the switch delay is $W = \frac{N}{\rho} \cdot \left(\frac{1}{TP} - 1\right)$.

Proof: By Proposition 1, $P_{ij}^1 = 1 - TP_{ij}$. Since $B = 1$, P_{ij}^1 is also the expectation of the number of packets stored in the buffer. Further, in the steady state, the probability that a packet arrives at the buffer in an arbitrary time slot is $\lambda_{ij} = \rho/N$, and by Definition 3, the probability that it arrives and is absorbed in the buffer is $(\rho/N)TP_{ij}$. Hence, applying Little's law on B_{ij} yields $W_{ij} = \frac{1-TP_{ij}}{\frac{\rho}{N} \cdot TP_{ij}}$. By symmetry, we obtain the required expression for the switch delay. ■

We now prove that for a CQ switch with $B = 1$ and a uniform Bernoulli i.i.d traffic pattern, every work-conserving scheduling algorithm yields the same throughput and delay. Therefore, analyzing the throughput of *any* work-conserving algorithm suffices to characterize all other work-conserving algorithms.

Theorem 1: For a CQ switch with $B = 1$ and uniform Bernoulli i.i.d traffic pattern, every work-conserving scheduling algorithm obtains the same throughput and delay.

Proof: Each output j provides services independently of other outputs. Therefore, without loss of generality, we can focus on an arbitrary column j, and consider its column throughput TP_{c_j}. Let $\mathcal{M}$ denote the Markov chain that represents the number of occupied buffers in column j before the arrival phase. Then $\mathcal{M}$ is a Markov chain over the state space $\mathcal{S} = \{0, \ldots, N-1\}$ with some transition matrix T.

Since each work-conserving scheduling algorithm provides the same number of services for a given state $s \in \mathcal{S}$, and the number of packets arrived at the empty crosspoint buffers is independent of the scheduling algorithm, then each work-conserving scheduling algorithm yields the same transition matrix T of the Markov chain $\mathcal{M}$. Further, $\mathcal{M}$ is ergodic. Therefore, each work-conserving scheduling algorithm also yields the same probability π_j^0 that all the buffers are empty after the arrival phase. By Proposition 2, it follows that TP_{c_j} is the same for each such scheduler, implying that all work-conserving algorithms obtain the same throughput. Further, by Proposition 3, they obtain the same delay as well. ■

B. Closed-Form Performance Results

We found that in a CQ switch with $B = 1$ and uniform traffic, all work-conserving schedules have the exact same performance. This enables us to provide a closed-form formula of the switch throughput and delay, by directly solving the Markov chain that characterizes the system.

Theorem 2: For a CQ switch with $B = 1$, uniform Bernoulli i.i.d arrivals of load ρ, and any work-conserving scheduling algorithm, the switch throughput TP and the switch delay W are

$$TP = \frac{1}{\rho}\left[1 - \frac{q}{1 + \sum_{m=1}^{N-1}\binom{N-1}{m} q^{-m} \prod_{j=1}^{m}\left(q^{-j}-1\right)}\right],$$

where $q = 1 - \frac{\rho}{N}$, and

$$W = \frac{N}{\rho} \cdot \left(\frac{1}{TP} - 1\right).$$

Proof: Let X_t be a random variable representing the number of occupied crosspoint buffers destined to some output j *before* the arrival phase of time-slot t. Since the scheduler is work-conserving and X_t is evaluated immediately after the departure phase of time-slot $t-1$, we cannot have $X_t = N$. Therefore, as in the proof of Theorem 1, X_t follows a Markov chain on state space $\mathcal{S} = \{0, \dots, N-1\}$. Further, let $p = \frac{\rho}{N}$ be the probability that a packet arrives at some input i and is destined for some output j at time-slot t, and let $q = 1 - p$. Then the state transition matrix of the Markov chain is provided by:

$$\begin{aligned} P_{mk} &= \Pr(X_t = k | X_{t-1} = m) \\ &= \begin{cases} \binom{N-m}{k-m+1} p^{k-m+1} q^{N-1-k} & (m \le k+1) \text{ and} \\ & \text{not } m = k = 0 \\ Npq^{N-1} + q^N & m = k = 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

This transition matrix results from simple probabilistic arguments. For instance, the first case intuitively reflects the fact that for X_t to increase by $k-m$ packets, it first needs to increase by $k-m+1$ packets before the column is served and a packet departs. This happens if a subset of $k-m+1$ crosspoints out of the $N-m$ empty ones receive a packet, and the crosspoints in the complementary subset do not receive any. Now, in order to determine the throughput of the switch, we focus on the steady-state probability vector. Specifically, let p_i be the steady-state probability to be in state $i \in \mathcal{S}' = \{0, \dots, N-1\}$. Then by definition,

$$p_i = \sum_{m=0}^{N-1} p_m P_{mi}, \quad \text{and} \quad \sum_{i=0}^{N-1} p_i = 1. \tag{2}$$

As in [40], we now compute p_i using the following z-transform of the sequence $\{p_0, p_1, ..., p_{N-1}\}$:

$$P(z) = \sum_{i=0}^{N-1} p_i z^{N-1-i} = \sum_{k=0}^{N-1} p_{N-1-k} z^k \tag{3}$$

Substituting Equation (2) in Equation (3),

$$\begin{aligned} P(z) &= \sum_{i=0}^{N-1}\sum_{m=0}^{N-1} p_m P_{mi} z^{N-1-i} \\ &= \sum_{m=0}^{N-1} p_m \sum_{i=0}^{N-1} P_{mi} z^{N-1-i} \end{aligned} \tag{4}$$

On the other hand, from Equation (1) and the Binomial Theorem, for $m \in \{0, \dots, N-1\}$,

$$\begin{aligned} &\textstyle\sum_{i=0}^{N-1} P_{mi} z^{N-1-i} \\ &= \begin{cases} (p+zq)^{N-m} & m \neq 0 \\ (p+zq)^N + q^N z^{N-1} - q^N z^N & m = 0 \end{cases} \\ &= \begin{cases} (1+(z-1)q)^{N-m} & m \neq 0 \\ (1+(z-1)q)^N - (z-1) q^N z^{N-1} & m = 0 \end{cases} \end{aligned}$$

Thus, inserting these equations into Equation (4), we get

$$\begin{aligned} P(z) &= -p_0 (z-1) q^N z^{N-1} \\ &\quad + \sum_{m=0}^{N-1} p_m (1+(z-1)q)^{N-m} \\ &= -p_0 (z-1) q^N \left[\sum_{m=0}^{N-1}\binom{N-1}{m}(z-1)^m\right] \\ &\quad + (1+(z-1)q)\, P(1+(z-1)q), \end{aligned} \tag{5}$$

where the last equality comes from the expansion of z^{N-1} and from the definition of $P(1+(z-1)q)$ (Equation (3)). Further, as defined in Equation (3), $P(z)$ is a polynomial of degree $N-1$, which can also be written as a polynomial of the same degree but in the neighborhood of $z = 1$:

$$P(z) = \sum_{i=0}^{N-1} \alpha_i (z-1)^i \tag{6}$$

Thus, using Equation (3), the Taylor series expansion yields coefficients α_i for $0 \le i \le N-1$:

$$\alpha_i = \frac{P(z)^{(i)}\big|_{z=1}}{i!} = \sum_{m=i}^{N-1}\binom{m}{i} p_{N-1-m} \tag{7}$$

We can now find that the N equalities in Equation (7) and the normalization condition in Equation (2) yield a set of $N+1$ equations with $N+1$ variables $(p_0, \alpha_0, \alpha_1, \dots, \alpha_{N-1})$.

For space reasons, we present the intermediate steps in [39] instead. The equations yield:

$$p_0 = \frac{1}{q^{N-1} + \sum_{m=1}^{N-1} \left(\binom{N-1}{m} q^{N-1-m} \prod_{j=1}^{m} (q^{-j} - 1) \right)}$$

Recall that p_0 is the probability that all the buffers are empty *before* the arrival phase, while π_j^0 is the same probability *after* the arrival phase, which happens if and only if all the buffers were empty before the arrival phase and no packet arrived. Thus $\pi_j^0 = p_0 \cdot q^N$, i.e.

$$\pi_j^0 = \frac{q}{1 + \sum_{m=1}^{N-1} q^{-m} \binom{N-1}{m} \prod_{j=1}^{m} (q^{-j} - 1)}, \tag{8}$$

and the throughput TP of the system is given by Proposition 2. Moreover, using Proposition 3, we also get the column delay. Finally, by symmetry, column and switch performances are equal. ■

The above closed-form expression can be used to compute two direct corollaries (refer to [39] for the proof). The first corollary presents the throughput of a 2×2 CQ switch.

Corollary 3: Using any work-conserving schedule, the throughput of a 2×2 CQ switch with $B = 1$ and a uniform traffic pattern of load $\rho = 1$ is $\frac{5}{6}$.

Likewise, the closed-form expression yields the following asymptotic result for CQ switches with large port count. Note that a similar result previously appeared in [10], albeit with an incomplete proof [41].

Corollary 4: Using any work-conserving schedule, the throughput of an $N \times N$ CQ switch with $B = 1$ and a uniform traffic pattern of load $\rho = 1$ goes to 1 as $N \to \infty$.

IV. LQF Scheduling

In the next three sections, we successively analyze three CQ switch scheduling algorithms: longest queue first (LQF), random, and exhaustive round-robin.

In an LQF-based CQ switch, each output schedules the longest queue in its column, resolving ties uniformly at random. In this section, we compare the performance of a CQ switch using LQF scheduling and the theoretically-optimal output-queued (OQ) switch with the same total buffer size.

A. Deterministic Guarantees

Consider a traffic pattern for which the throughput of the OQ switch goes to one when B goes to infinity. We want to prove that the throughput of the LQF-based CQ switch goes to one as well. We show it deterministically, using the fact that LQF tends to equalize the buffer occupancy. For space reasons, the proofs appear in [39].

Theorem 5: For any traffic pattern, the throughput of an LQF-based CQ switch with crosspoint buffer size B is at least the throughput of an OQ switch with output buffer size $2B-1$ and at most the throughput of an OQ switch with output buffer size NB.

Corollary 6: For any traffic pattern, whenever defined, the throughput limits of LQF-based CQ switches and OQ switches are the same when B goes to infinity.

B. Statistical Model

In practice, under a large variety of traffic patterns, LQF-based CQ switches behave quite similarly to OQ switches with the *same total buffer size*. The intuition is that as the buffer size increases, the effects of the static buffer partitioning disappear because of the LQF equalizing effects. This is intuitive for uniform Bernoulli i.i.d. arrivals, but appears to hold as well for non-uniform Bernoulli i.i.d. as well as bursty long-range-dependent traffic.

Consequently, we model the statistical behavior of LQF-based CQ switches using OQ-based models. In particular, in the framework of *large deviations theory* [42], we model the overflow probability $\bar{P}_j$ of LQF-based CQ switches in column j as being *equal* to that of the OQ switch, i.e. to the probability that the OQ buffer occupancy Q_j^{OQ} of output j exceeds some given value b assuming infinite buffers and the same arrival pattern. Then we obtain the following model for $\bar{P}_j$ (we prove the OQ overflow probability result in [39] using large-deviations theory [42]).

Theorem 7: The overflow probability $\bar{P}_j$ in column j of an LQF-based CQ switch with total buffer size b and Bernoulli i.i.d. arrivals of admissible rate matrix $[\lambda_{ij}]$ can be modeled by $\bar{P}_j \approx e^{-\theta^* b}$, where

$$\begin{aligned} \theta^* &= -\lim_{b\to\infty} \frac{1}{b} \log(Pr(Q_j^{OQ} > b)) \\ &= \left\{ \sup_{\theta>0} \theta : \sum_{i=1}^{N} \log\left(1 - \lambda_{ij} + \lambda_{ij} e^{\theta}\right) < \theta \right\}. \end{aligned} \tag{9}$$

This model is compared against simulations in Section VII.

V. Random Scheduling

Our objective is to model the performance of the random scheduling algorithm, in which each output picks uniformly at random a buffer to serve.

A first approach is to model the states of all the different crosspoint buffers as *independent*. However, this approximation is too crude, because the buffer occupancies in the same column j are typically correlated.

Therefore, we use a finer model based on conditional probabilities. For any given input i and output j, we study the conditional probability that buffer B_{ij} is empty in the current time-slot given that all the buffers $\{B_{1j}, B_{2j}, \ldots, B_{(i-1)j}\}$ from the inputs with smaller indices are empty as well. We then multiply all these conditional probabilities to get the probability that all the buffers in column j are empty given that the first one is empty, i.e. given that any buffer is empty by symmetry. We obtain the following model (refer to [39] for the proof):

Proposition 4: The throughput TP of a CQ switch with a buffer size B and a random scheduling algorithm under uniform Bernoulli i.i.d. traffic arrival can be modeled by

$$TP = \left(1 - \left(\frac{\rho}{N}\frac{1-s}{s}\right)^B\right) P^0, \tag{10}$$

using the solution to the following set of three equations with three unknowns (P^0, s, s_k):

$$\begin{cases} P^0 = \frac{\rho - 1 + \prod_{k=0}^{N-1}\left[P^0\left(1-\frac{\rho}{N}\right)\left(1-\frac{\rho}{N}+\frac{\rho}{N}\frac{s_k}{s}\right)\right]}{\rho\cdot\left(\frac{\rho}{N}\frac{1-s}{s}\right)^B}, \\ P^0 \cdot \left[\sum_{i=0}^{B-1}\left(\frac{\frac{\rho}{N}(1-s)}{\left(1-\frac{\rho}{N}\right)s}\right)^i + \left(\frac{\frac{\rho}{N}(1-s)}{s}\right)^B\right] = 1 \\ s_k = \left(1-\frac{\rho}{N}\right)^k \frac{1-\left(P^0\left(1-\frac{\rho}{N}\right)\right)^{N-k}}{\left(1-P^0\left(1-\frac{\rho}{N}\right)\right)(N-k)} + \left(1-\left(1-\frac{\rho}{N}\right)^k\right)\cdot \\ \quad \frac{\left(1-P^0\left(1-\frac{\rho}{N}\right)\right)(N-k+1)-1+\left(P^0\left(1-\frac{\rho}{N}\right)\right)^{(N-k+1)}}{\left(1-P^0\left(1-\frac{\rho}{N}\right)\right)^2(N-k+1)(N-k)} \end{cases}$$

VI. Exhaustive Round-Robin Scheduling

We now want to model the throughput of a CQ switch operating with the *exhaustive round-robin* algorithm. This algorithm considers all crosspoint buffers in a round-robin order and, upon finding a non-empty buffer, services it until it is empty. Thus, this algorithm corresponds to a *polling system* with zero *switch-over times* [43]. As later shown in simulations, it typically has a higher packet throughput for variable-size packets than a round-robin algorithm, because it tends to read together all the cells of the same packet.

A natural throughput model relies on evaluating the average steady-state cycle time t_0, i.e. the average number of time-slots taken by the round-robin algorithm to visit all the crosspoints in a given column. In the steady state, we would expect t_0 to be equal to the expected number of packets arriving at this column within t_0 time-slots. Unfortunately, this first approach to modeling throughput does not work, because sometimes all the column buffers are empty. In this case, the zero switch-over times cause an infinite number of cycles with zero cycle time, thus biasing the average cycle time.

To overcome this problem, for each column j we define an arbitrary buffer (e.g., the last buffer) and assume that the scheduler always pauses for one time-slot after servicing it, independently of the column state, thus virtually removing the zero switch-over time. We then compute the steady-state cycle time t_0', and subtract this time-slot back at the end to get a more accurate model using $t_0 = t_0' - 1$. We obtain the following result (the proof is in [39]).

Proposition 5: The throughput TP of a CQ switch with a buffer size B and an exhaustive round-robin scheduling algorithm under a uniform Bernoulli i.i.d. traffic arrival of load ρ can be modeled by

$$TP = \frac{B - \sum_{k=0}^{B-1}(B-k)\begin{pmatrix} t_0 \\ k \end{pmatrix}\left(\frac{\rho}{N}\right)^k\left(1-\frac{\rho}{N}\right)^{(t_0-k)}}{\frac{\rho\cdot t_0}{N}}$$

where $t_0 = t_0' - 1$, and t_0' is the solution of the following fixed-point equation (which is solved numerically):

$$t_0' = N\left(B - \sum_{k=0}^{B-1}(B-k)\begin{pmatrix} t_0' \\ k \end{pmatrix}\left(\frac{\rho}{N}\right)^k\left(1-\frac{\rho}{N}\right)^{t_0'-k}\right) + 1.$$

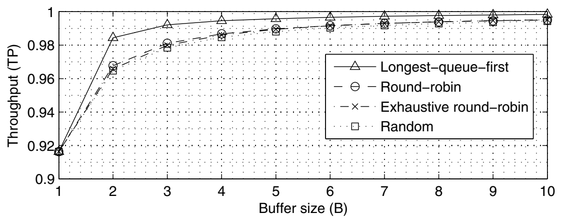


Fig. 2. Throughput of a 32×32 CQ switch under uniform traffic

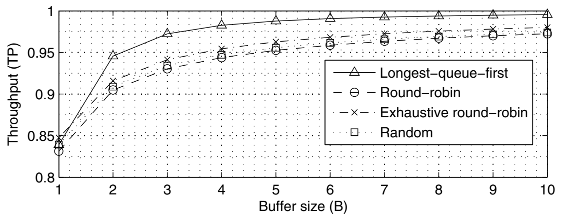


Fig. 3. Throughput of a 32×32 CQ switch under non-uniform log-diagonal traffic

VII. Simulations

We now compare the theorems obtained in former sections against simulation results. We show that the CQ switch can obtain a throughput close to 100% with moderate crosspoint buffer sizes, and in particular that LQF-based CQ switches behave closely to OQ switches.

A. Scheduler Performance Comparison

We first want to check Theorem 1 in Section III, which states that every work-conserving scheduling algorithm obtains the same throughput in a CQ switch with $B = 1$ and uniform Bernoulli i.i.d traffic pattern.

Fig. 2 and Fig. 3 compare the throughput of four different work-conserving scheduling algorithms on a 32×32 CQ switch with different buffer sizes, under Bernoulli traffic arrival of rate $\rho = 1$. In Fig. 2 the incoming traffic is uniform, while in Fig. 3 the incoming traffic is non-uniform and log-diagonal: for each input $1 \leq i \leq 31$ and output j, $\lambda_{ij} = 2^{-i}$ and $\lambda_{32,j_0} = 2^{-31}$ [44]. Each simulation was performed 10 times with 10^5 time-slots each time, yielding a maximum standard deviation per run of $1.9 \cdot 10^{-3}$, thus the plot should be fairly accurate.

When $B = 1$ and the traffic is uniform, the obtained throughput appears indeed to be the same for all these work-conserving scheduling algorithms, thus confirming Theorem 1.

Furthermore, when $B > 1$, the LQF scheduler clearly outperforms all other schedulers, regardless of the buffer size. The throughput of the three other scheduling algorithms (random, round-robin, and exhaustive round-robin) appears relatively close.

B. Overflow Probability of LQF Scheduling

We now want to check Theorem 7 of Section IV, which models the overflow rate of the LQF-based CQ switch using the overflow rate of the OQ switch.

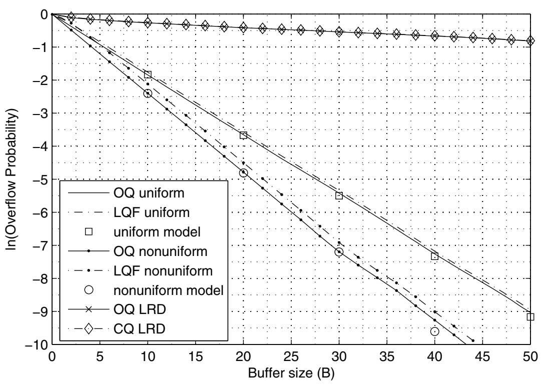


Fig. 4. Overflow probability for an 8×8 LQF-based CQ switch and an OQ switch under uniform and non-uniform Bernoulli and uniform bursty traffic.

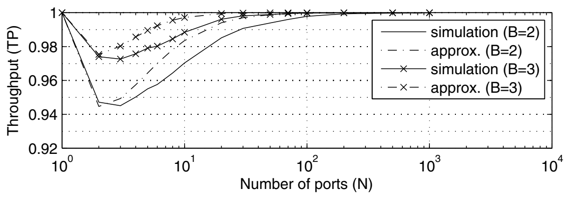


Fig. 5. Throughput of the random scheduling algorithm.

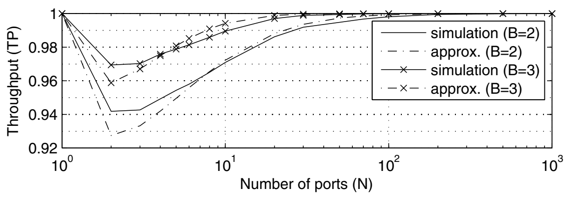


Fig. 6. Throughput of the exhaustive round-robin scheduling algorithm.

Fig. 4 displays the overflow probability on an arbitrary column j of buffers of an 8×8 LQF-based CQ switch. The simulations were run under three different types of arrival processes. First, under uniform Bernoulli i.i.d. arrival traffic of load $\rho = 0.99$, with a run-time of 10^9 time-slots. Then, under non-uniform log-diagonal Bernoulli i.i.d. arrival traffic with a run-time of 10^9 time-slots, using for each input $1 \leq i \leq 7$ and output j, $\lambda_{ij} = 0.99 \cdot 2^{-i}$ and $\lambda_{8,j} = 0.99 \cdot 2^{-7}$ [44]. Finally, under bursty long-range-dependent (LRD) traffic with a run-time of 10^6 time-slots, with load $\lambda_{ij} = \frac{0.99}{8}$ and Hurst parameter 0.6, using the algorithm proposed in [45].

The simulation results confirm that the overflow rates of the LQF-based CQ switch and of the OQ switch behave in the same way. In particular, the overflow probability formula in Theorem 7 matches well our simulation results, both in the uniform and non-uniform cases. Surprisingly, the overflow rates of the LQF-based CQ switch and of the OQ switch also behave similarly for bursty long-range-dependent traffic arrivals.

C. Random and Exhaustive Round-Robin Scheduling

We now check the models of Proposition 4 in Section V and Proposition 5 in Section VI on the performances of the random and exhaustive round-robin scheduling algorithm.

Fig. 5 and Fig. 6 compare the simulation results with the predicted results using $B = 2$ and $B = 3$, using both scheduling algorithms. They plot the throughput against the number of ports, under uniform Bernoulli i.i.d. arrivals with $\rho = 0.9$. The plots should be fairly accurate: each simulation was performed 10 times using 10^4 time-slots each time, with a maximum standard deviation of $3.7 \cdot 10^{-3}$ per run.

Both models clearly reflect the non-monotonic performance variations of the scheduling algorithms. The exhaustive round-robin model is a bit closer to simulated values across the wide range of port number values, while the random model appears to be too optimistic on the throughput.

D. Real-Life Traces

We evaluate now the performance of the CQ switch in a slightly more realistic setting, using real-life traces with variable-size packets.

When packets have variable size, they are usually segmented at the inputs into fixed-size cells and later reassembled again at the destination. Thus, a variable-size packet is *fully lost* if one or more of its cells are lost. Further, large packets have more chances of having lost cells. To take this into account, we consider the *packet throughput*, denoted PTP, i.e. the ratio of the number of bits of fully-delivered packets by the number of bits of arrived packets. Of course, the value of PTP is *worse* than the value of the throughput TP, since losing one cell of a large packet counts as if all cells were lost.

Fig. 7 plots the packet throughput of a 32×32 CQ switch as a function of the crosspoint buffer size for different scheduling algorithms. These include a new algorithm, *Exhaustive LQF (Longest Queue First)*, which upon finding a crosspoint with the longest queue, services it until it is empty. In the simulations, each input was trace-driven by a different OC-48 CAIDA link trace [46], and the switch external rate was normalized to the lowest needed rate so as to provide admissibility. Therefore, the CQ switch effectively operated at an average load of 1, well above expected loads in the Internet.

The simulation results show that the CQ switch is still able to achieve good performance in this more realistic trace-based setting, especially for $B \geq 64$ cells. As expected, the LQF scheduler obtains the best results, followed by the exhaustive LQF and round-robin algorithms, and then the non-exhaustive round-robin algorithm. This simulation confirms that even under high load, when using $B = 64$ cells of 64 bytes per crosspoint, a 32Mb SRAM-based CQ switch can achieve good performance results.

VIII. Conclusion

In this paper, we introduced the CQ switch and showed that it can switch packets without permanently relying on the current states of the input linecards, the output linecards, and the complex centralized scheduler. We explained how it can be readily implemented on a single chip. We finally illustrated how the CQ switch would provide a performance comparable

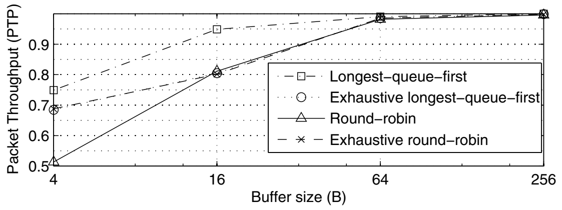


Fig. 7. Packet throughput of a 32×32 CQ switch under trace-based traffic.

to that of an ideal output-queued switch under most statistical traffic patterns, using analysis as well as simulations based on synthetic and real-life arrival traces.

Acknowledgment

This work was partly supported by the European Research Council Starting Grant n° 210389, the Alon Fellowship, the ATS-WD Career Development Chair, and the Loewengart Research Fund. We would also like to thank Ran Ginosar and Paolo Giaccone for useful discussions.

References

[1] 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, Jun. 1999.

[2] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE Infocom*, vol. 2, pp. 556–564, 2000.

[3] F. Abel, C. Minkenberg, R. P. Luijten, M. Gusat, and I. Iliadis, "A four-terabit packet switch supporting long round-trip times," *IEEE Micro*, vol. 23, no. 1, pp. 10–24, 2003.

[4] A. P. J. Engbersen, "Prizma switch technology," *IBM Journal of Research and Development*, vol. 47, no. 2-3, pp. 195–210, 2003.

[5] M. Nabeshima, "Performance evaluation of a combined input- and crosspoint-queued switch," *IEICE Trans. Comm.*, vol. 83, no. 3, pp. 737–741, 2000.

[6] R. Rojas-Cessa, E. Oki, and H. J. Chao, "CIXOB-k: combined input-crosspoint-output buffered packet switch," *IEEE Globecom*, vol. 4, pp. 2654–2660, Nov. 2001.

[7] R. Magill, C. Rohrs, and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE J. Select. Areas Commun.*, vol. 21, no. 4, pp. 606–615, May 2003.

[8] S.-T. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," *IEEE Infocom*, vol. 2, pp. 981–991, Mar. 2005.

[9] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, "Variable packet size buffered crossbar (CICQ) switches," *IEEE ICC*, vol. 2, pp. 1090–1096, Jun. 2004.

[10] M. Lin and N. McKeown, "The throughput of a buffered crossbar switch," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 465–467, May 2005.

[11] D. Pan and Y. Yang, "Localized asynchronous packet scheduling for buffered crossbar switches," *ACM/IEEE ANCS*, pp. 153–162, 2006.

[12] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *IEEE Infocom*, 2006.

[13] P. Giaccone, E. Leonardi, and D. Shah, "Throughput region of finite-buffered networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 251–263, Feb. 2007.

[14] S. He, S. Sun, H.-T. Guan, Q. Zheng, Y. Zhao, and W. Gao, "On guaranteed smooth switching for buffered crossbar switches," *IEEE/ACM Trans. Networking*, vol. 16, no. 3, pp. 718–731, 2008.

[15] G. F. Georgakopoulos, "Buffered cross-bar switches, revisited: Design steps, proofs and simulations towards optimal rate and minimum buffer memory," *IEEE/ACM Trans. Networking*, vol. 16, no. 16, pp. 1340–1351, Dec. 2008.

[16] P. Giaccone and E. Leonardi, "Asymptotic performance limits of switches with buffered crossbars," *IEEE Trans. Inform. Theory*, vol. 54, no. 2, pp. 595–607, Feb. 2008.

[17] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," *ACM SIGCOMM*, vol. 32, no. 4, pp. 251–264, 2002.

[18] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," *ACM SIGCOMM*, vol. 33, no. 4, pp. 189–200, 2003.

[19] S. Iyer and N. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 314–324, 2003.

[20] F. Abel, C. Minkenberg, I. Iliadis, A. P. J. Engbersen, M. Gusat, F. Gramsamer, and R. P. Luijten, "Design issues in next-generation merchant switch fabrics," *IEEE/ACM Trans. Networking*, vol. 15, no. 6, pp. 1603–1615, 2007.

[21] C. Minkenberg, R. P. Luijten, F. Abel, W. Denzel, and M. Gusat, "Current issues in packet switch design," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 119–124, 2003.

[22] P. Goli and V. Kumar, "Performance of a crosspoint buffered ATM switch fabric," *IEEE Infocom*, pp. 426–435, 1992.

[23] H. Tomonaga, N. Matsuoka, Y. Kato, and Y. Watanabe, "High-speed switching module for a large capacity ATM switching system," *IEEE Globecom*, vol. 1, pp. 123–127, Dec. 1992.

[24] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimoto, "Integrated services packet network using bus matrix switch," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1284–1292, Oct. 1987.

[25] E. Del Re and R. Fantacci, "Performance evaluation of input and output queueing techniques in atm switching systems," *IEEE Trans. Commun.*, vol. 41, no. 10, pp. 1565–1575, Oct. 1993.

[26] International Technology Roadmap for Semiconductors (ITRS), "Executive summary," 2007.

[27] IEEE P802.3ae 10Gb/s Ethernet Task Force, "XAUI standard," 2001.

[28] H. Attiya, D. Hay, and I. Keslassy, "Packet-mode emulation of output-queued switches," *ACM SPAA*, pp. 138–147, 2006.

[29] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet-mode scheduling in input-queued cell-based switches," *IEEE/ACM Trans. Networking*, vol. 10, no. 5, pp. 666–678, Oct. 2002.

[30] Y. Ganjali, A. Keshavarzian, and D. Shah, "Input queued switches: cell switching vs. packet switching," *IEEE Infocom*, vol. 3, pp. 1651–1658, Mar. 2003.

[31] N. McKeown, "Sizing router buffers," *Presentation*, Stanford, 2006.

[32] G. Shrimali, I. Keslassy, and N. McKeown, "Designing packet buffers with statistical guarantees," *IEEE Hot Interconnects*, 2004.

[33] J. Garcia-Vidal, M. March, L. Cerda, J. Corbal, and M. Valero, "A DRAM/SRAM memory scheme for fast packet buffers," *IEEE Trans. Comput.*, May 2006.

[34] S. Iyer, R. R. Kompella, and N. McKeown, "Designing packet buffers for router linecards," *IEEE/ACM Trans. Networking*, vol. 16, no. 3, pp. 705–717, 2008.

[35] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM*, vol. 34, no. 4, pp. 281–292, 2004.

[36] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," *Next Generation Internet Networks*, pp. 173–180, Apr. 2005.

[37] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," *IEEE Infocom*, pp. 1–11, Apr. 2006.

[38] M. Shifrin and I. Keslassy, "Modeling TCP in small-buffer networks," *Networking*, pp. 667–678, 2008.

[39] Y. Kanizo, D. Hay, and I. Keslassy, "The crosspoint-queued switch," Comnet, Technion, Israel, Technical Report TR08-04, 2008. [Online]. Available: http://comnet.technion.ac.il/~isaac/papers.html

[40] S. K. Bose, *An Introduction to Queueing Systems*. Springer, 2002. [Online]. Available: http://home.iitk.ac.in/~skb/qbook/MG1-N.PDF

[41] Personal Communication.

[42] A. Ganesh, N. O'Connell, and D. Wischik, *Big Queues*. Springer, 2004.

[43] H. Takagi, *Analysis of polling systems*. MIT Press, 1986.

[44] D. Shah, P. Giaccone, and B. Prabhakar, "An efficient randomized algorithm for input-queued switch scheduling," *IEEE Hot Interconnects*, Aug. 2001.

[45] R. G. Clegg and M. Dodson, "Markov chain-based method for generating long-range dependence," *Phys. Rev. E*, vol. 72, no. 2, Aug. 2005.

[46] "Cooperative association for internet data analysis (CAIDA)." [Online]. Available: http://www.caida.org/