A Dynamic Frame Sizing Algorithm for CICQ Switches with 100% Throughput

Cheng-Shang Chang, Yu-Hao Hsu, Jay Cheng, and Duan-Shin Lee
Institute of Communications Engineering
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
Email: cschang@ee.nthu.edu.tw
yhhsu@gibbs.ee.nthu.edu.tw
jcheng@ee.nthu.edu.tw
lds@cs.nthu.edu.tw

***Abstract*—A Combined Input and Crosspoint Queueing (CICQ) switch is a switch that has both buffers at the crosspoints of the switch fabric and buffers at the inputs. Inspired by the fixed frame based algorithm for an input-buffered switch in [19] and the smooth scheduling algorithm for a CICQ switch in [11], in this paper we propose using a *dynamic* frame sizing algorithm for a CICQ switch. It is formally shown that such a CICQ switch indeed achieves 100% throughput for certain Poisson-like traffic models. This is done *without* using the framed Birkhoff-von Neumann decomposition needed in [19]. Moreover, such a CICQ switch only requires a two-cell buffer at each crosspoint when there is only *unicast* traffic. Unlike input-buffered switches, the dynamic frame sizing algorithm also achieves 100% throughput in the setting of *multicast* traffic. This is done at the cost of increasing the buffer size at each crosspoint.**

***Index Terms*—Combined input and crosspoint queueing, input-buffered switches, 100% throughput.**

I. Introduction

Recently, there are re-surged interests in the crosspoint buffered switches due to the advances of VLSI technologies (see e.g., [18], [20], [13], [17], [7], [24], [25], [9], [10], [26], [11] and reference therein). With the N^2 hardware complexity for an $N \times N$ crosspoint buffered switch, it was once regarded as an unscalable switch architecture and can only be used for constructing small switches or as building blocks for multistage switches [1]. Even so, it is a common belief that it is possible to build a 128×128 crosspoint buffered switch fabric, capable of holding one cell at each crosspoint, in one single chip with the current technology [9].

One of the most salient features of crosspoint buffered switches is the decoupling property of the scheduling policies between inputs and outputs. Unlike an unbuffered crossbar, where packets transmitted from an input must be received by an output at the same time, buffers at the crosspoints provide additional flexibility for designing scheduling policies in the inputs/outputs of crosspoint buffered switches. Clearly, if one has an unlimited buffer at each crosspoint, then one has perfect decoupling between inputs and outputs. However, in practice, one can only have a limited buffer size at each crosspoint and thus requires further buffering at the inputs of a crosspoint buffered switch. A switch that has both buffers at the crosspoints and at the inputs is then called a Combined Input and Crosspoint Queueing (CICQ) switch.

There are many interesting results on CICQ switches. For instance, it was shown in [13] that 100% throughput can be achieved for uniform traffic under the Longest Queue First (LQF) policy at the inputs of a crosspoint buffered switch with a one-cell buffer at each crosspoint. The 100% throughput result was extended in [25] for all admissible traffic by considering the shortest crosspoint buffer first (SCBF) policy that requires global queue length information in each time slot. With a speedup factor of 2, it was shown in [17], [9] that one can use a crosspoint buffered switch to emulate an output-buffered switch for certain scheduling polices. This is further generalized to the setting with variable length packets in [22]. Using the Dual Maximum Weight First algorithm, it is shown in [12] that one can achieve ρ-efficiency in CICQ switches, where the crosspoint buffer size depends on the maximum arrival rate ρ. In addition to the issue of throughput, providing fairness in crosspoint buffered switches was also addressed in [7], [26]. More recently, it was shown in [11] that CICQ switches can also provide rate guarantees with a two-cell buffer at each crosspoint. During the process of writing this paper, Prof. H. J. Chao was kindly to notify us the paper [21], where a practical 100% throughput algorithm using Hamiltonian walk was proposed for CICQ switches.

In this paper, we proposed another approach to achieve 100% throughput in an $N \times N$ CICQ switch. Our approach is inspired by the frame based algorithm in [19] for an input-buffered switch with virtual output queues. As such, we will also start from proving the 100% throughput in input-buffered switches in Section II and then extend it to CICQ switches in Section III. The key difference between the frame based algorithm in [19] and ours is that we adopt a *dynamic* frame sizing algorithm, where the frame size is different in each frame. We note that such a dynamic algorithm was previously mentioned in the simulation section in [19] without analysis. The key idea of our dynamic frame sizing algorithm is to operate the system in frames and clear up all the backlogs that appear in the input buffers at the beginning of a frame before the end of that frame. As such, the backlogs at the beginning

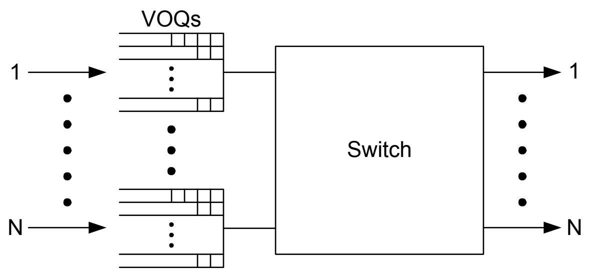

Fig. 1. An $N \times N$ input-buffered switch with VOQs.

of a frame is then bounded above by the arrivals during the previous frame, and a packet that arrives in one frame will depart in the next frame. As long as the traffic is admissible, the expected frame size is shown to have an $O(\log N)$ bound. In comparison with the fixed size frame algorithm in [19], our dynamic frame sizing algorithm does not need to know the traffic load for choosing the appropriate frame size.

To clear up the backlogs in input buffers, one needs to carry out the framed Birkhoff-von Neumann decomposition described in [5]. Such a decomposition has an $O(N^{2.5}T)$ complexity, where T is the frame time. To avoid using the framed Birkhoff-von Neumann decomposition, a smooth scheduling policy with $O(\log N)$ complexity at each input/crosspoint buffer was developed in [11] for an CICQ switch with a two-cell buffer at each crosspoint. The key idea of the smooth scheduling policy in [11] is to send packets from an input buffer to a crosspoint buffer with a constant rate and then retrieve them with the same rate. In the ideal fluid case, there is no need for buffer. However, for the packetized version, one needs a two-cell buffer at each crosspoint. With the help of the smooth scheduling policy in [11], we can then extend the 100% throughput result to CICQ switches under our dynamic frame sizing algorithm for both the unicast traffic in Section III and the multicast traffic in Section IV.

To summarize, an $N \times N$ CICQ switch with the dynamic frame sizing algorithm has the following nice features: (i) it achieves 100% throughput for certain Poisson-like traffic models, (ii) it only requires a finite buffer at each crosspoint, (iii) it is also applicable to the setting with multicast traffic, and (iv) there is no need to carry out the framed Birkhoff-von Neumann decomposition in [19].

II. Dynamic Frame Sizing Algorithms for Input-buffered Switches

In this section, we introduce our dynamic frame sizing algorithm and show that it achieves 100% throughput in an input-buffered switch. Throughput this paper, we consider the usual discrete-time setting by assuming that packets are of the same size and that time is slotted in every link so that a packet can be transmitted within a time slot. Also, we assume that every switch is started from an empty system at time 0. Packets arrive from time 1 onward (including time 1).

Consider an $N \times N$ input-buffered switch (see Figure 1). At each input, there are N virtual output queues (VOQs), indexed from $1, 2, \ldots, N$. When a packet destined for the j^{th} output arrives at an input, it is placed in the j^{th} VOQ of that input. Let $\mathbf{p}(t) = (p_{i,j}(t))$ be the permutation matrix for the connection pattern at time t, i.e., the i^{th} input is connected to the j^{th} output at time t if $p_{i,j}(t) = 1$. Let $a_{i,j}(t)$ be the number of packets that arrive at the i^{th} input and are destined for the j^{th} output at time t, and $q_{i,j}(t)$ be the number of packets in the j^{th} VOQ of the i^{th} input at time t (at the end of the t^{th} time slot). Then we have the following governing equation for such an input-buffered switch:

$$q_{i,j}(t+1) = (q_{i,j}(t) + a_{i,j}(t+1) - p_{i,j}(t+1))^+, \qquad (1)$$

where $x^+ = \max(0, x)$.

Now we describe the dynamic sizing algorithm that determines the connection patterns for the input-buffered switch. The dynamic frame sizing algorithm is a framed based algorithm. However, the frame size is not fixed and it is determined by the minimum clearance time in [19], [23] at the beginning of every frame. The minimum clearance time is the minimum amount of time to clear all the packets currently stored in the VOQs if there are no future arrivals. Specifically, suppose there are initially $q_{i,j}$ packets in the j^{th} VOQ of the i^{th} input. Then using the (framed) Birkhoff-von Neumann decomposition described in [5], the minimum clearance time, denoted by T, is simply the maximum of the maximum column sum and the maximum row sum of the matrix $Q = (q_{i,j})$, i.e.,

$$T = \max\Big[\max_{1 \le i \le N} \sum_{j=1}^{N} q_{i,j}, \max_{1 \le j \le N} \sum_{i=1}^{N} q_{i,j}\Big].$$

At the beginning of a frame, the minimum clearance time is used as the new frame size if not every VOQ is empty.

Let T_n be the size of the n^{th} frame and $\tau_n = \sum_{i=1}^{n-1} T_i$. Thus, $\tau_n + 1$ is the first time slot of the n^{th} frame and τ_{n+1} is the last time slot of the n^{th} frame. Let $A_{i,j}(t) = \sum_{s=1}^{t} a_{i,j}(s)$ be the cumulative number of packets that arrive at the i^{th} input by time t and are destined for the j^{th} output. Recursively expanding from (1), we have

$$\begin{aligned} q_{i,j}(\tau_{n+1}) &= \max\Big[\max_{0 \le s \le T_n - 1}[A_{i,j}(\tau_{n+1}) \\ &- A_{i,j}(\tau_{n+1} - s) - \sum_{t=\tau_{n+1}-s+1}^{\tau_{n+1}} p_{i,j}(t)], \\ q_{i,j}(\tau_n) &+ A_{i,j}(\tau_{n+1}) - A_{i,j}(\tau_n) - \sum_{t=\tau_n+1}^{\tau_{n+1}} p_{i,j}(t)\Big]. \end{aligned} \qquad (2)$$

From the definition of the minimum clearance time, we can use the framed Birkhoff-von Neumann decomposition to set up the permutation matrices $\mathbf{p}(t)$ so that by the end of the n^{th} frame those packets left from the previous frame are cleared, i.e.,

$$\sum_{t=\tau_n+1}^{\tau_{n+1}} p_{i,j}(t) \ge q_{i,j}(\tau_n).$$

Using this in (2) implies that the number of packets left at the end of the n^{th} frame can only come from the new arrivals in that frame, i.e.,

$$q_{i,j}(\tau_{n+1}) \leq A_{i,j}(\tau_{n+1}) - A_{i,j}(\tau_n). \qquad (3)$$

The inequality in (3) is the key inequality for the dynamic frame sizing algorithm and it will be used to show (in Theorem 3 later) that the dynamic frame sizing algorithm indeed achieves 100% throughput under the assumptions (A1-3) described below.

Now we introduce the assumptions on our traffic model. Let $A_i^I(t) = \sum_{j=1}^N A_{i,j}(t)$, $i = 1, 2, \ldots, N$, be the cumulative number of arrivals at the i^{th} input by time t, and $A_j^O(t) = \sum_{i=1}^N A_{i,j}(t)$, $j = 1, 2, \ldots, N$, be the cumulative number of packets destined for the j^{th} output by time t.

(A1) The arrival processes $\{a_{i,j}(t), t = 1, 2, \ldots, \}$, $i, j = 1, 2, \ldots, N$, are adapted to a common filtration $\mathcal{F}_t$, i.e., $a_{i,j}(t)$ is a $\mathcal{F}_t$-measurable random variable.

(A2) For $s, t \geq 0$, there are finite-valued functions $\rho(\theta)$ and $\sigma(\theta)$ (independent of s and t) such that

$$\frac{1}{\theta} \log E(e^{\theta(A_i^I(t+s) - A_i^I(s))} | \mathcal{F}_s) \leq \rho(\theta)t + \sigma(\theta), \quad i = 1, 2, \ldots, N, \qquad (4)$$

and

$$\frac{1}{\theta} \log E(e^{\theta(A_j^O(t+s) - A_j^O(s))} | \mathcal{F}_s) \leq \rho(\theta)t + \sigma(\theta), \quad j = 1, 2, \ldots, N. \qquad (5)$$

(A3) There exists some $\theta > 0$ and $\epsilon > 0$ such that

$$\rho(\theta) \leq 1 - \epsilon. \qquad (6)$$

The assumptions (A1-3) were previously introduced in [3] to show the sample path large deviations of intree networks. Examples that satisfy (A1-3) (see Examples 2.5, 2.6. and 2.7 in [3]) are finite state Markov arrival processes, renewal processes, and autoregressive processes. The function $\rho(\theta)$ in all these examples can be derived from the maximum of the following limits

$$\rho_i^I(\theta) = \lim_{t \to \infty} \frac{1}{\theta t} \log \mathsf{E}[e^{\theta A_i^I(t)}], \quad i = 1, 2, \ldots, N,$$
$$\rho_j^O(\theta) = \lim_{t \to \infty} \frac{1}{\theta t} \log \mathsf{E}[e^{\theta A_j^O(t)}], \quad j = 1, 2, \ldots, N.$$

The functions $\rho_i^I(\theta)$ and $\rho_j^O(\theta)$ are known as the effective bandwidth functions in the literature (see e.g., [14] and the references therein). It is well known that the effective bandwidth functions are increasing in θ and they approach their mean arrival rates when $\theta \to 0$. Thus, as long as the mean arrival rate at any input and the mean arrival rate at any output are strictly less than 1, one can always find a $\theta > 0$ and $\epsilon > 0$ so that (A3) is satisfied for finite state Markov arrival processes, renewal processes, and autoregressive processes in [3].

Example 1: **(Poisson traffic model)** To better illustrate the physical meanings of the assumptions (A1-3), let us consider the simplest case that $\{a_{i,j}(t), t = 1, 2, \ldots, \}$, $i, j = 1, 2, \ldots, N$, are independent sequences of i.i.d. Poisson random variables. Let $\lambda_{i,j}$ be the rate of $\{a_{i,j}(t), t = 1, 2, \ldots, \}$, $i, j = 1, 2, \ldots, N$, and these rates satisfy the following no overbooking condition:

$$\sum_{j=1}^N \lambda_{i,j} \leq \rho < 1, \quad i = 1, 2, \ldots, N, \qquad (7)$$

and

$$\sum_{i=1}^N \lambda_{i,j} \leq \rho < 1, \quad j = 1, 2, \ldots, N. \qquad (8)$$

The condition in (7) and (8) says that the total arrival rate at any input/output is less than 1.

As the sum of independent Poisson random variables is still a Poisson random variable, we know that $A_i^I(t+s) - A_i^I(s)$ is a Poisson random variable with parameter $\sum_{j=1}^N \lambda_{i,j} t$ and this random variable is independent of everything else. Let $\mathcal{F}_t$ be the filtration generated by the arrival processes by time t, i.e., $\{a_{i,j}(s), s = 1, 2, \ldots, t, i, j = 1, 2, \ldots, N\}$. Thus, we have

$$\begin{aligned} & \frac{1}{\theta} \log E(e^{\theta(A_i^I(t+s) - A_i^I(s))} | \mathcal{F}_s) \\ = & \frac{1}{\theta} \log E(e^{\theta(A_i^I(t+s) - A_i^I(s))}) \\ = & \frac{1}{\theta} \sum_{j=1}^N \lambda_{i,j} t (e^\theta - 1) \\ \leq & \frac{e^\theta - 1}{\theta} \rho t, \quad i = 1, 2, \ldots, N, \end{aligned} \qquad (9)$$

where we use (7) in the last inequality. Similarly, we also have

$$\begin{aligned} & \frac{1}{\theta} \log E(e^{\theta(A_j^O(t+s) - A_j^O(s))} | \mathcal{F}_s) \\ = & \frac{1}{\theta} \log E(e^{\theta(A_j^O(t+s) - A_j^O(s))}) \\ \leq & \frac{e^\theta - 1}{\theta} \rho t, \quad j = 1, 2, \ldots, N. \end{aligned} \qquad (10)$$

Thus, we can choose

$$\rho(\theta) = \rho \frac{e^\theta - 1}{\theta}$$

and $\sigma(\theta) = 0$ for (A1) and (A2). Furthermore, if we choose $\epsilon = (1 - \rho)/2$ and θ^* from the unique positive solution of the equation

$$\frac{e^\theta - 1}{\theta} = \frac{1 + \rho}{2\rho}. \qquad (11)$$

Then

$$\rho(\theta^*) = (1 + \rho)/2 = 1 - (1 - \rho)/2 = 1 - \epsilon,$$

and (A3) is satisfied.

To gain further intuition on (11), note that when ρ is very close 1, θ^* is very close to 0 in (11). As such, we may approximate the function $e^\theta - 1$ by $\theta + \theta^2/2$ and use that to approximate θ^* by $(1 - \rho)/\rho$.

Example 2: **(Bernoulli traffic model)** Another commonly used traffic model in the literature is the Bernoulli traffic model. Let $a_i^I(t)$ be the number of packets that arrive at the i^{th} input at time t. Assume that $\{a_i^I(t), t = 1, 2, \ldots\}$, $i = 1, 2, \ldots, N$, are independent sequences of i.i.d. Bernoulli random variables with parameters λ_i, $i = 1, 2, \ldots, N$, i.e., with probability λ_i, there is a packet arriving at the i^{th} input in a time slot and this is independent of everything else. Also, a packet that arrives at the i^{th} input is randomly chosen for the j^{th} output with probability $\alpha_{i,j}$, where $\alpha_{i,j} \geq 0$ and $\sum_{j=1}^N \alpha_{i,j} = 1$. Such a choice is independent of everything else. For the Bernoulli traffic model, $\{a_{i,j}(t), t = 1, 2, \ldots\}$ is still a sequence of i.i.d. Bernoulli random variables with parameter $\lambda_i\alpha_{i,j}$. As the no overbooking condition in Example 1, we assume

$$\lambda_i = \sum_{j=1}^N \lambda_i\alpha_{i,j} \leq \rho < 1, \quad i = 1, 2, \ldots, N, \tag{12}$$

and

$$\sum_{i=1}^N \lambda_i\alpha_{i,j} \leq \rho < 1, \quad j = 1, 2, \ldots, N. \tag{13}$$

As in Example 1, let $\mathcal{F}_t$ be the filtration generated by the arrival processes by time t, i.e., $\{a_{i,j}(s), s = 1, 2, \ldots, t, i, j = 1, 2, \ldots, N\}$. Since $\{a_i^I(t), t = 1, 2, \ldots\}$ is a sequence of i.i.d. Bernoulli random variables with parameters λ_i, we have

$$\begin{aligned} & \frac{1}{\theta}\log E(e^{\theta(A_i^I(t+s)-A_i^I(s))}|\mathcal{F}_s) \\ = \; & \frac{1}{\theta}\log E(e^{\theta(A_i^I(t+s)-A_i^I(s))}) \\ = \; & \frac{1}{\theta}\log(\lambda_i e^\theta + 1 - \lambda_i)^t \\ = \; & \frac{t}{\theta}\log(\lambda_i e^\theta + 1 - \lambda_i) \\ \leq \; & \frac{e^\theta - 1}{\theta}\lambda_i t, \end{aligned} \tag{14}$$

where we use $\log(1 + x) \leq x$ in the last inequality. In conjunction with the assumption in (12),

$$\frac{1}{\theta}\log E(e^{\theta(A_i^I(t+s)-A_i^I(s))}|\mathcal{F}_s) \leq \frac{e^\theta - 1}{\theta}\rho t \tag{15}$$

On the other hand, $\{a_{i,j}(t), t = 1, 2, \ldots\}$, $i = 1, 2, \ldots, N$, are *independent* sequences of i.i.d. Bernoulli random variables with parameter $\lambda_i\alpha_{i,j}$, $i = 1, 2, \ldots, N$. Thus,

$$\begin{aligned} & \frac{1}{\theta}\log E(e^{\theta(A_j^O(t+s)-A_j^O(s))}|\mathcal{F}_s) \\ = \; & \frac{1}{\theta}\log E(e^{\theta(A_j^O(t+s)-A_j^O(s))}) \\ = \; & \frac{1}{\theta}\log E(e^{\theta(\sum_{i=1}^N \sum_{\tau=s+1}^{t+s} a_{i,j}(\tau))}) \\ = \; & \frac{t}{\theta}\sum_{i=1}^N \log(\lambda_i\alpha_{i,j}e^\theta + 1 - \lambda_i\alpha_{i,j}) \\ \leq \; & \frac{t}{\theta}\sum_{i=1}^N \lambda_i\alpha_{i,j}(e^\theta - 1), \end{aligned} \tag{16}$$

where we once again use $\log(1+x) \leq x$ in the last inequality. It then follows from (13) that

$$\frac{1}{\theta}\log E(e^{\theta(A_j^O(t+s)-A_j^O(s))}|\mathcal{F}_s) \leq \frac{e^\theta - 1}{\theta}\rho t, \;\; j = 1, 2, \ldots, N. \tag{17}$$

As the two inequalities in (15) and (17) are the same as those in (9) and (10) in Example 1, we can choose

$$\rho(\theta) = \rho\frac{e^\theta - 1}{\theta}$$

and $\sigma(\theta) = 0$ for (A1) and (A2). Furthermore, if we choose $\epsilon = (1-\rho)/2$, then θ^* can be chosen from the unique positive solution of (11) so that (A3) is satisfied. This shows that we can use the same bound derived for the Poisson traffic model in this example.

Our main result of this section is the following bound on the expected frame size. The proof is given in Appendix A.

Theorem 3: Assume that the input traffic satisfies (A1-3). Then under the dynamic frame sizing algorithm, we have for $n \geq 1$

$$\mathsf{E}[T_n] \leq \max\Big[1, (\log(2N)/\theta + \sigma(\theta))/\epsilon\Big]. \tag{18}$$

The bound in Theorem 3 implies 100% throughput of the input-buffered switch for finite state Markov arrival processes, renewal processes, and autoregressive processes in [3]. Now using Theorem 3 for the Poisson traffic model in Example 1 and the Bernoulli traffic model in Example 2, we show in the following corollary that the expected frame size for these two traffic models is $O(\log N)$.

Corollary 4: For the Poisson traffic model in Example 1 and the Bernoulli traffic model in Example 2, the expected frame size is bounded above by $\max[1, 2\log(2N)/(1-\rho)\theta^*]$, where θ^* is the unique positive solution of (11).

When ρ is very close to 1, we may approximate θ^* by $(1-\rho)/\rho$ and the expected frame size in Corollary 4 is then roughly bounded by $2\rho\log(2N)/(1-\rho)^2$.

III. From Input-buffered Switches to CICQ Switches

The problem of implementing the dynamic frame sizing algorithm in an input-buffered switch is the needed framed Birkhoff-von Neumann decomposition at the beginning time slot of every frame. The need for carrying out the framed Birkhoff-von Neumann decomposition can be avoided by using Combined Input and Crosspoint Queueing (CICQ) switches as previously proposed in [11] for providing guaranteed rate services in CICQ switches. Like an input-buffered switch described in the previous section, a CICQ switch also has N VOQs at each input. Instead of using a simple crossbar switch fabric, a CICQ switch uses a *buffered* crossbar switch fabric (see Figure 2). When a packet destined for the j^{th} output is sent from the i^{th} input, it is stored in the $(i, j)^{th}$ crosspoint buffer and then retrieved by the j^{th} output. With the help of crosspoint buffers, a packet in a CICQ switch needs not be sent (from its input) and retrieved (by its output) at the same

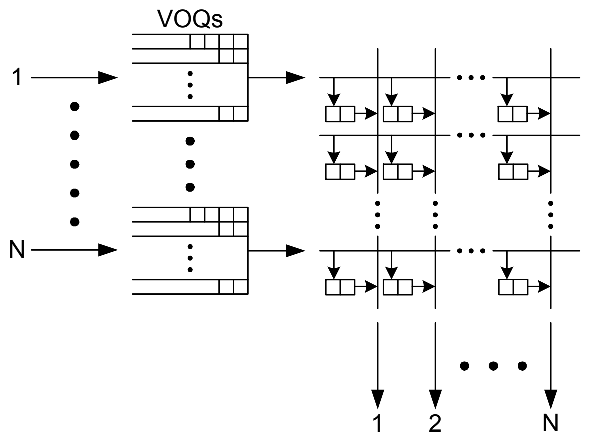

Fig. 2. An $N \times N$ CICQ switch.

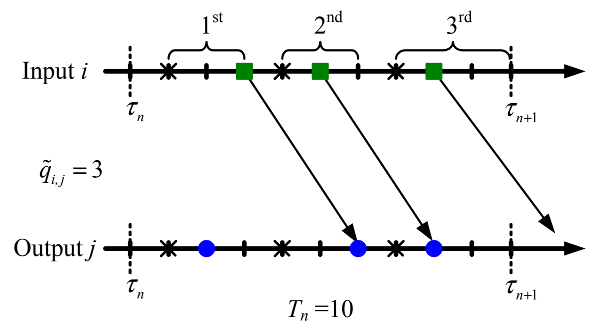

Fig. 3. An illustrating example of the scheduling algorithm for $\tilde{q}_{i,j} = 3$ and $T_n = 10$.

time. This relaxes the matching constraint in an input-buffered switch that requires packets to be sent and retrieved at the same time. To limit the buffer size at each crosspoint, the key idea is to send and retrieve packets with the same "rate" in every frame.

In this section, we will show how the dynamic frame sizing algorithm can be used in a CICQ switch that only requires each crosspoint buffer to hold at most two packets. To do this, we need the following lemma on the earliest deadline first (EDF) policy for scheduling deterministic flows of packets with constant arrival rates. The proof of Lemma 5 is given in Appendix B.

Lemma 5: Consider feeding N flows of packets to a buffered link. The link is capable of transmitting a packet in a time slot. Packets within each flow are assigned nondecreasing deadlines. The EDF policy is the policy that serves the packet with the earliest deadline among all the packets in the buffer in every time slot (ties are broken arbitrarily). Suppose that the cumulative number of flow i packets that arrive by time t is $\lceil r_i t \rceil$ for all t (with some $r_i \geq 0$) and the number of flow i packets that have deadlines not later than t is $\lceil r_i(t+1) \rceil - 1$ for all t, $i = 1, 2, \dots, N$. If $\sum_{i=1}^{N} r_i \leq 1$, then every packet is served not later than its deadline under the EDF policy.

Remark 6: Note that if the arrival rate of flow i in Lemma 5 is positive, i.e., $r_i > 0$, then the k^{th} flow i packet arrives at time $1 + \lfloor (k-1)/r_i \rfloor$. To see this, note that $\lceil r_i t \rceil = k$ if and only if $k \geq r_i t > k - 1$. Thus,

$$\frac{k}{r_i} \geq t > \frac{k-1}{r_i}.$$

The least integer that satisfies the above inequality is $1 + \lfloor (k-1)/r_i \rfloor$. Following the same argument, the k^{th} flow i packet is assigned with the deadline $\lfloor k/r_i \rfloor$. Then the result in Lemma 5 ensures that the k^{th} flow i packet, arriving at time $1 + \lfloor (k-1)/r_i \rfloor$, is served not later than $\lfloor k/r_i \rfloor$. As such, flow i behaves as if it were a constant rate flow with rate r_i.

Now we describe our implementation of the dynamic frame sizing algorithm in a CICQ switch.

(S1) *Determine the size of a new frame:* this is the same as that for an input-buffered switch in the previous section. Specifically, let $q_{i,j}(t)$ be the number of packets in the j^{th} VOQ of the i^{th} input at time t and τ_n be the last time slot of the $(n-1)^{th}$ frame (and $\tau_n + 1$ be the beginning time slot of the n^{th} frame). If there are packets in one of the VOQs of the N inputs, the size of the n^{th} frame is set to be

$$T_n = \max\Big[\max_{1 \leq i \leq N} \sum_{j=1}^{N} q_{i,j}(\tau_n), \max_{1 \leq j \leq N} \sum_{i=1}^{N} q_{i,j}(\tau_n)\Big]. \quad (19)$$

On other hand, if all the VOQs of the N inputs are empty, then T_n is set to be 1 and we skip the rest of the steps.

(S2) *Schedule packets in the VOQs with the rates proportional to their sizes at the beginning of a frame:* consider the i^{th} input. Let $\tilde{q}_{i,j} = q_{i,j}(\tau_n)$. For the j^{th} VOQ of the i^{th} input, $j = 1, 2, \dots, N$, $\tilde{q}_{i,j}$ tokens are generated. The k^{th} token for the j^{th} VOQ, $k = 1, 2, \dots, \tilde{q}_{i,j}$, is assigned with the eligible time $\tau_n + 1 + \lfloor (k-1)T_n/\tilde{q}_{i,j} \rfloor$ and the deadline $\tau_n + \lfloor kT_n/\tilde{q}_{i,j} \rfloor$. In each time slot of the n^{th} frame, i.e., the time interval $[\tau_n + 1, \tau_n + T_n]$, the eligible token with the earliest deadline is selected and removed from the pool of tokens. Ties are broken arbitrarily. If the selected token is for the j^{th} VOQ and the j^{th} VOQ is not empty, then the head-of-line packet of the j^{th} VOQ is sent to the switch fabric in that time slot. Clearly, the complexity for the earliest deadline policy at each input is $O(\log N)$ in every time slot as one only needs to find the token with the earliest deadline among (at most) N eligible tokens in every time slot.

(S3) *Schedule packets in the crosspoint buffers with the same rates of their corresponding VOQs:* consider the j^{th} output. Let $\tilde{q}_{i,j} = q_{i,j}(\tau_n)$. For the $(i,j)^{th}$ crosspoint buffer, $i = 1, 2, \dots, N$, $\tilde{q}_{i,j}$ tokens are generated. The k^{th} token for $(i,j)^{th}$ crosspoint buffer, $k = 1, 2, \dots, \tilde{q}_{i,j}$, is assigned with the eligible time $\tau_n + 1 + \lfloor (k-1)T_n/\tilde{q}_{i,j} \rfloor$ and the deadline $\tau_n + \lfloor kT_n/\tilde{q}_{i,j} \rfloor$. In each time slot of the n^{th} frame, the eligible token with the earliest deadline is selected and removed from the pool of tokens. Ties are broken arbitrarily. If the selected token is for the $(i,j)^{th}$ crosspoint buffer and the $(i,j)^{th}$ crosspoint buffer is not empty, then the head-of-line packet of the $(i,j)^{th}$ crosspoint buffer is sent to the j^{th} output in that time slot.

The earliest deadline first (EDF) policy used for scheduling packets in both the VOQs and the crosspoint buffers was originally proposed in [11] for avoiding Birkhoff-von Neumann decomposition needed in guaranteed rate services. Note from (19) that

$$\sum_{j=1}^{n} \frac{\tilde{q}_{i,,j}}{T_n} \le 1, \quad i = 1, 2, \ldots, N, \tag{20}$$

$$\sum_{i=1}^{n} \frac{\tilde{q}_{i,,j}}{T_n} \le 1, \quad j = 1, 2, \ldots, N. \tag{21}$$

From Lemma 5 and Remark 6, we know that every token is served not later than its deadline. Specifically, the k^{th} token for the j^{th} VOQ of the i^{th} input (in the n^{th} frame) is served in the interval $[\tau_n + 1 + \lfloor (k-1)T_n/\tilde{q}_{i,j} \rfloor, \tau_n + \lfloor kT_n/\tilde{q}_{i,j} \rfloor]$. So is the k^{th} token for the $(i,j)^{th}$ crosspoint buffer. In Figure 3, we provide an illustrating example for the timing diagram of the scheduling algorithm. In this example, we assume that $\tilde{q}_{i,j} = 3$ and $T_n = 10$. There are three tokens generated in this frame for both the j^{th} VOQ of the i^{th} input and the $(i,j)^{th}$ crosspoint buffer. The eligible times of these three tokens are marked with "$\times$". The time slots marked with "$\Box$" are the time slots when the tokens for the j^{th} VOQ of the i^{th} input are served. The time slots marked with "$\circ$" are the time slots when the tokens for the $(i,j)^{th}$ crosspoint buffer are served. In this example, the packet carried by the first (resp. second) token for the j^{th} VOQ of the i^{th} input is served by the second (resp. third) token for the $(i,j)^{th}$ crosspoint buffer.

We note that there may be no eligible tokens in a time slot. When this happens, that time slot is wasted. To improve the system performance, one can include an additional step after (S1) by enlarging the queue length matrix with the von Neumann algorithm (see e.g., [5]). Specifically, one can find an integer-valued matrix $\tilde{Q} = (\tilde{q}_{i,j})$ such that $\tilde{q}_{i,j} \ge q_{i,j}(\tau_n)$ and

$$\sum_{j=1}^{n} \tilde{q}_{i,,j} = T_n, \quad i = 1, 2, \ldots, N, \tag{22}$$

$$\sum_{i=1}^{n} \tilde{q}_{i,,j} = T_n, \quad j = 1, 2, \ldots, N. \tag{23}$$

Clearly, the normalized matrix $\tilde{Q}/T_n$ is a doubly stochastic matrix (where all the row sums and column sums are equal to 1) and this will make sure that there are always eligible tokens in every time slot.

For this scheduling algorithm, we can infer the following facts: (P1) *Bounded expected frame size:* in the n^{th} frame, there are exactly $\tilde{q}_{i,j}$ tokens served for the j^{th} VOQ of the i^{th} input. As $\tilde{q}_{i,j} \ge q_{i,j}(\tau_n)$, the inequality in (3) holds. As such, Theorem 3 is still holds under the traffic assumptions in (A1-3). As a direct consequence of Theorem 3, we know that the expected frame size is bounded above by a finite constant when the arrival processes are finite state Markov arrival processes, renewal processes, and autoregressive processes in [3] as long as the mean arrival rate at any input and the mean arrival rate at any output are strictly less than 1.

(P2) *Maximum number of packets in a crosspoint buffer:* index the tokens continuously from time 0 by removing the boundaries of frames. The packet (if there is any) carried by the k^{th} token for the j^{th} VOQ of the i^{th} input is sent to the j^{th} output either by the k^{th} token or the $(k+1)^{th}$ token for the $(i,j)^{th}$ crosspoint buffer. The worst case is then the k^{th} token for the j^{th} VOQ of the i^{th} input is served *after* the k^{th} token for the $(i,j)^{th}$ crosspoint buffer is served, and the $(k+1)^{th}$ token for the j^{th} VOQ of the i^{th} input is served *before* the $(k+1)^{th}$ token for the $(i,j)^{th}$ crosspoint buffer is served. In such a case, the two packets carried by the k^{th} token and the $(k+1)^{th}$ token for the j^{th} VOQ of the i^{th} input are stored in the $(i,j)^{th}$ crosspoint buffer until the $(k+1)^{th}$ token for the $(i,j)^{th}$ crosspoint buffer is served (see e.g., Figure 3). Thus, each crosspoint buffer only needs to hold at most two packets. This is one of the major results reported in [11].

From (P1) and (P2), we conclude that the dynamic frame sizing algorithm achieves 100% throughput for a CICQ switch when the arrival processes satisfy (A1-3). This includes finite state Markov arrival processes, renewal processes, and autoregressive processes in [3] with the mean arrival rate at any input and the mean arrival rate at any output being strictly less than 1.

Note that in (S3) a time slot is wasted when the selected token is for the $(i,j)^{th}$ crosspoint buffer and the $(i,j)^{th}$ crosspoint buffer is *empty*. To improve the system performance, one can further allow the scheduler to select another non-empty crosspoint buffer as described below.

(S3*) *Schedule packets in the crosspoint buffers with the same rates of their corresponding VOQs:* follow the same scheduling algorithm as in (S3). In addition to this, we keep an *input* pointer in each output. When the selected token is for the $(i,j)^{th}$ crosspoint buffer and the $(i,j)^{th}$ crosspoint buffer is *empty*, then the nonempty crosspoint buffer that is closest to the input pointer is selected, and the head-of-line packet of the selected crosspoint buffer is sent to the j^{th} output. The input pointer of the j^{th} output is then advanced *clockwise* to one location beyond the selected crosspoint buffer.

IV. CICQ Switches With Multicast Traffic

It is known (see e.g., [2]) that an input-buffered switch (with an unbuffered crossbar) cannot achieve 100% throughput with multicast traffic, even when fanout splitting is allowed. The main objective of this section is to show that our dynamic frame sizing algorithm also achieves 100% throughput for CICQ switches with multicast traffic. However, this is at the cost of increasing the buffer size at each crosspoint.

Specifically, suppose that there are L_i (unicast and multicast) flows from the i^{th} input, $i = 1, 2, \ldots, N$. Denote by $A_{i,\ell}$ the ℓ^{th} flow from the i^{th} input and by $A_{i,\ell}(t)$ the cumulative number of flow $A_{i,\ell}$ packets that arrive by time t. Let $S_{i,\ell}$ be the fanout set of flow $A_{i,\ell}$, i.e., the set of outputs of flow $A_{i,\ell}$. Also, let $S^*_{i,j} = \{(i,\ell) : j \in S_{i,\ell}\}$ be the set of flows from the i^{th} input that contains the j^{th} output as one of its destinations.

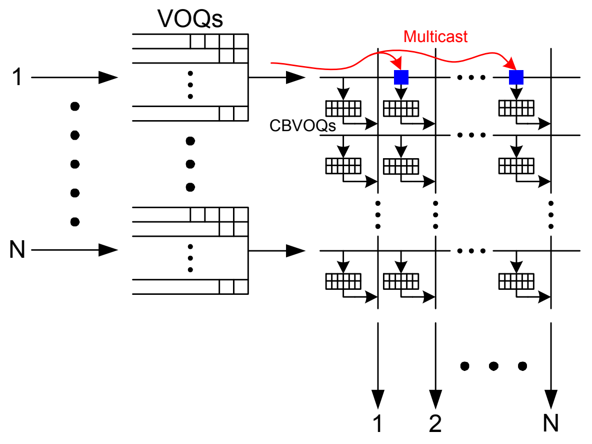

Fig. 4. An $N \times N$ CICQ switch for multicast traffic.

To cope with multicast traffic, we need to enhance the CICQ switch architecture as shown in Figure 4. At the i^{th} input, there are L_i VOQs, indexed from $1, 2, \ldots, L_i$. A flow $A_{i,\ell}$ packet is placed into the ℓ^{th} VOQ when it arrives. In addition to this, there are $|S^*_{i,j}|$ CBVOQs in the crosspoint buffer between the i^{th} input and the j^{th} output, with each CBVOQ corresponding to one flow of packets in $S^*_{i,j}$. When a flow $A_{i,\ell}$ packet is sent from the i^{th} input to the switch fabric, the packet is multicast to all the crosspoint buffers connected to one of the outputs in $S_{i,\ell}$ (see Figure 4). Specifically, if $S_{i,\ell}$ contains the j^{th} output, then a copy of the flow $A_{i,\ell}$ packet is placed in one of the $|S^*_{i,j}|$ CBVOQs that corresponds to flow $A_{i,\ell}$.

To determine the size of a new frame, let $q_{i,\ell}(t)$ be the number of flow $A_{i,\ell}$ packets stored in the ℓ^{th} VOQ of the i^{th} input at time t and τ_n be the last time slot of the $(n-1)^{th}$ frame. The size of the n^{th} frame is set to be

$$T_n = \max\Big[\max_{1\le i\le N}\sum_{\ell=1}^{L_i} q_{i,\ell}(\tau_n), \max_{1\le j\le N}\sum_{(i,\ell)\in S^*_j} q_{i,\ell}(\tau_n)\Big]. \tag{24}$$

Scheduling packets in the VOQs of the inputs and packets in the CBVOQs of crosspoint buffers are basically the same as (S2) and (S3) with $q_{i,\ell}(\tau_n)$ tokens being generated for flow $A_{i,\ell}$ in the n^{th} frame. As in Section III, those packets stored in input buffers at time τ_n will be cleared by the end of the n^{th} frame. Thus, we still have

$$q_{i,\ell}(\tau_{n+1}) \le A_{i,\ell}(\tau_{n+1}) - A_{i,\ell}(\tau_n) \tag{25}$$

for all $\ell = 1, 2, \ldots, L_i$ and $i = 1, 2, \ldots, N$. Following the same argument for Theorem 3, one can then show that the dynamic frame sizing algorithm also achieves 100% throughput for multicast traffic, and there are at most two packets in every CBVOQ of a crosspoint buffer.

V. Simulations

In this section, we perform various simulations for our dynamic frame sizing algorithm specified in (S1), (S2), and (S3*). In our simulations, we consider the *uniform* Bernoulli traffic model, i.e., $\lambda_i = \rho$ for all $i = 1, 2, \ldots, N$ and $\alpha_{i,j} = 1/N$ for all $i, j = 1, 2, \ldots, N$, in Example 2. Each simulation is run with 10^7 time slots. As shown in Table I of [6], under the unform Bernoulli traffic model the expected delay for the output-buffered switch is $\frac{N-1}{N}\frac{\rho}{2(1-\rho)}$ and the expected delay for the load balanced Birkhoff-von Neumann is $\frac{N-1}{2(1-\rho)}$. We then compare the measured average delay from our simulations with the expected delay in the output-buffered switch and the expected delay in the load balanced Birkhoff-von Neumann switch for $\rho = 0.5$ in Figure 5 and $\rho = 0.9$ in Figure 6, respectively. In each of these two figures, the curve marked with "VOQ Delay" is the average packet delay through the VOQs in an input buffer, and the curve marked with "System Delay" is the average packet delay through the CICQ switch under our dynamic frame sizing algorithm. As shown in these two figures, the output-buffered switch has the best delay performance. Also, the curves for the delay in the output-buffered switches are almost independent of the switch size N as the delay is $O(1)$. As the frame size is $O(\log N)$ in Theorem 3 under the dynamic frame sizing algorithm in a $N \times N$ CICQ switch, we expect that the delay in such a system is also $O(\log N)$ (though a formal proof like the one in [19] is needed), and the simulation results in Figure 5 and Figure 6 seem to confirm this. Even though the delay in the load balanced Birkhoff-von Neumann switch is $O(N)$, the delay performance of the CICQ switch under the dynamic frame sizing algorithm is not always better than that of the load balanced Birkhoff-von Neumann switch. In fact, as shown in Figure 6, the delay of the load balanced Birkhoff-von Neumann switch is smaller than that in the CICQ switch when N is small.

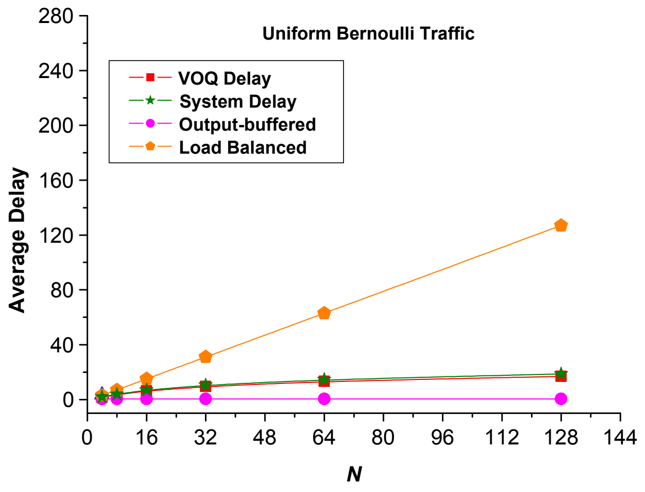

Fig. 5. Delay comparison for the uniform Bernoulli traffic model with $\rho = 0.5$.

VI. Conclusion

Inspired by the fixed frame based algorithm for an input-buffered switch in [19] and the smooth scheduling algorithm for a CICQ switch in [11], in this paper we proposed using a *dynamic* frame sizing algorithm for a CICQ switch. We showed that such a CICQ switch indeed achieves 100%

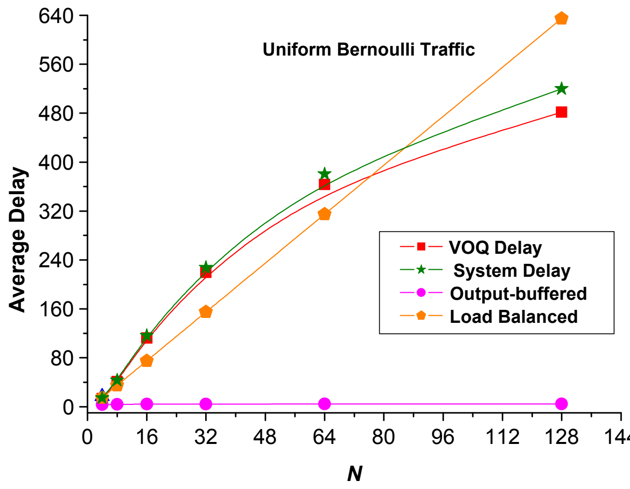

Fig. 6. Delay comparison for the uniform Bernoulli traffic model with $\rho = 0.9$.

throughput for certain Poisson-like traffic models. Such a CICQ switch does not need to carry out the framed Birkhoff-von Neumann decomposition in [19], and it only requires a two-cell buffer at each crosspoint when there is only unicast traffic. The dynamic frame sizing algorithm can be further extended to the setting with multicast traffic. Our preliminary result shows that it can also be used in the setting with long propagation delay. There are several problems that require further study: (i) variable length packets, (ii) long range dependent input [15], (iii) networks of CICQ switches, and (iv) the use of network coding [16].

Appendix A

In this appendix, we prove Theorem 3.

We first show (18). Note from the definition of the minimum clearance time that

$$T_{n+1} = \max\Big[\max_{1\le i\le N}\sum_{j=1}^N q_{i,j}(\tau_{n+1}),\ \max_{1\le j\le N}\sum_{i=1}^N q_{i,j}(\tau_{n+1})\Big]. \tag{26}$$

For $\theta > 0$, we then have

$$\begin{aligned} e^{\theta T_{n+1}} &= \max\Big[\max_{1\le i\le N}\exp(\theta\sum_{j=1}^N q_{i,j}(\tau_{n+1})),\\ &\qquad \max_{1\le j\le N}\exp(\theta\sum_{i=1}^N q_{i,j}(\tau_{n+1}))\Big]\\ &\le \sum_{i=1}^N \exp(\theta\sum_{j=1}^N q_{i,j}(\tau_{n+1}))\\ &\qquad + \sum_{j=1}^N \exp(\theta\sum_{i=1}^N q_{i,j}(\tau_{n+1})), \end{aligned} \tag{27}$$

where we use the inequality $\max[x_1, x_2] \le x_1 + x_2$ for $x_1, x_2 \ge 0$. Using (3) yields

$$\begin{aligned} e^{\theta T_{n+1}} &\le \sum_{i=1}^N \exp\Big(\theta\sum_{j=1}^N (A_{i,j}(\tau_{n+1}) - A_{i,j}(\tau_n))\Big)\\ &\qquad + \sum_{j=1}^N \exp\Big(\theta\sum_{i=1}^N (A_{i,j}(\tau_{n+1}) - A_{i,j}(\tau_n))\Big)\\ &= \sum_{i=1}^N \exp\Big(\theta(A_i^I(\tau_{n+1}) - A_i^I(\tau_n))\Big)\\ &\qquad + \sum_{j=1}^N \exp\Big(\theta(A_j^O(\tau_{n+1}) - A_j^O(\tau_n))\Big) \end{aligned} \tag{28}$$

Since T_n is determined by $q_{i,j}(\tau_n)$'s that in turn are determined by the arrival processes up to time τ_n, we know that T_n is $\mathcal{F}_{\tau_n}$-measurable. From (4) and (6), we then have

$$\begin{aligned} &\sum_{i=1}^N \mathsf{E}\Big[\exp\Big(\theta(A_i^I(\tau_{n+1}) - A_i^I(\tau_n))\Big)|\mathcal{F}_{\tau_n}\Big]\\ &\le \sum_{i=1}^N \exp\Big(\theta(\rho(\theta)T_n + \sigma(\theta))\Big)\\ &\le N e^{\theta(1-\epsilon)T_n} e^{\theta\sigma(\theta)},\ a.s. \end{aligned} \tag{29}$$

Similarly, we have from (5) and (6) that

$$\begin{aligned} &\sum_{j=1}^N \mathsf{E}\Big[\exp\Big(\theta(A_j^O(\tau_{n+1}) - A_j^O(\tau_n))\Big)|\mathcal{F}_{\tau_n}\Big]\\ &\le N e^{\theta(1-\epsilon)T_n} e^{\theta\sigma(\theta)},\ a.s. \end{aligned} \tag{30}$$

Using (29) and (30) in (28) yields

$$\mathsf{E}[e^{\theta T_{n+1}}|\mathcal{F}_{\tau_n}] \le 2N e^{\theta(1-\epsilon)T_n} e^{\theta\sigma(\theta)},\ a.s. \tag{31}$$

Taking expectations on both sides of (31) yields

$$\mathsf{E}[e^{\theta T_{n+1}}] \le 2N e^{\theta\sigma(\theta)} \mathsf{E}[e^{\theta(1-\epsilon)T_n}]. \tag{32}$$

This then implies

$$\log \mathsf{E}[e^{\theta T_{n+1}}] \le \log(2N) + \theta\sigma(\theta) + \log \mathsf{E}[e^{\theta(1-\epsilon)T_n}]. \tag{33}$$

Since $\phi(\theta) = \log \mathsf{E}[e^{\theta T_n}]$ is a convex function in θ (see e.g., [4], Proposition 7.1.8), we have

$$\begin{aligned} \log \mathsf{E}[e^{\theta(1-\epsilon)T_n}] &\le \epsilon \log \mathsf{E}[e^{0\cdot T_n}] + (1-\epsilon)\log \mathsf{E}[e^{\theta T_n}]\\ &= (1-\epsilon)\log \mathsf{E}[e^{\theta T_n}]. \end{aligned} \tag{34}$$

Using (34) in (33) yields

$$\log \mathsf{E}[e^{\theta T_{n+1}}] \le \log(2N) + \theta\sigma(\theta) + (1-\epsilon)\log \mathsf{E}[e^{\theta T_n}]. \tag{35}$$

Since $T_1 = 1$, it is easy to verify (18) from induction by using the inequality in (35).

Now we use (18) to show the delay bound in Theorem 3. Since $e^{\theta x}$ is a convex function in x, it follows from Jensen's inequality that

$$\mathsf{E}[T_n] \le \frac{1}{\theta}\log \mathsf{E}[e^{\theta T_n}] \le \max\Big[1, \frac{\frac{\log(2N)}{\theta} + \sigma(\theta)}{\epsilon}\Big]. \tag{36}$$

Appendix B

In this appendix, we prove Lemma 5.

Let $A_i(t)$ be the cumulative number of flow i packets that arrive by time t and $N_i(t)$ be the number of flow i packets that have deadlines not later than t. Thus, $A_i(t) = \lceil r_i t \rceil$ and $N_i(t) = \lceil r_i(t+1) \rceil - 1$. According to Theorem 2.3.17 in [4], it suffices to verify that for every subset S of $\{1, 2, \ldots, N\}$

$$\sum_{i \in S} N_i(t) \leq \min_{0 \leq s \leq t} \Big[\sum_{i \in S} A_i(s) + (t-s) \Big]. \tag{37}$$

To see this, note that for all $0 \leq s \leq t$

$$\begin{aligned}
&\sum_{i \in S} N_i(t) - \sum_{i \in S} A_i(s) \\
= \quad & \sum_{i \in S} (\lceil r_i(t+1) \rceil - 1) - \sum_{i \in S} \lceil r_i s \rceil \\
= \quad & \sum_{i \in S} (\lceil r_i(t+1) \rceil - \lceil r_i s \rceil) - |S| \\
\leq \quad & \sum_{i \in S} (\lceil r_i(t+1) - r_i s \rceil) - |S| \\
\leq \quad & \lceil \sum_{i \in S} r_i(t+1-s) \rceil + (|S|-1) - |S| \\
= \quad & \lceil \sum_{i \in S} r_i(t+1-s) \rceil - 1,
\end{aligned} \tag{38}$$

where we use $\lceil a \rceil \geq \lceil a+b \rceil - \lceil b \rceil$ in the first inequality and $\lceil a \rceil + \lceil b \rceil \leq \lceil a+b \rceil + 1$ in the second inequality. Since it is assumed that $\sum_{i=1}^{N} r_i \leq 1$, it then follows that for all $0 \leq s \leq t$

$$\begin{aligned}
\sum_{i \in S} N_i(t) - \sum_{i \in S} A_i(s) \quad & \leq \quad \lceil \sum_{i=1}^{N} r_i(t+1-s) \rceil - 1 \\
& \leq \quad t+1-s-1 = t-s.
\end{aligned} \tag{39}$$

Thus, we have for all $0 \leq s \leq t$

$$\sum_{i \in S} N_i(t) \leq \sum_{i \in S} A_i(s) + (t-s), \tag{40}$$

and this leads to (37).

References

[1] H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1091–1103, September 1989.

[2] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 465–477, June 2003.

[3] C.-S. Chang, "Sample path large deviations and intree networks," *Queueing Systems*, vol. 20, pp. 7–36, March 1995.

[4] C.-S. Chang, *Performance Guarantees in Communication Networks*, London: Springer-Verlag, 2000.

[5] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'00)*, Tel Aviv, Israel, March 26–30, 2000, pp. 1614–1623.

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

[7] N. Chrysos and M. Katevenis, "Weighted fairness in buffered crossbar scheduling," in *Proceedings IEEE Workshop on High Performance Switching and Routing (HPSR'03)*, Torino, Italy, June 24–27, 2003, pp. 17–22.

[8] N. Chrysos and M. Katevenis, "Scheduling in non-blocking buffered three-stage switching fabrics," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'06)*, Barcelona, Spain, April 23–29, 2006, pp. 1–13.

[9] S.-T. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'05)*, Miami, FL, USA, March 13–17, 2005, pp. 981–991.

[10] Z. Dong and R. Rojas-Cessa, "Long round-trip time support with shared-memory crosspoint buffered packet switch," in *Proceedings IEEE Symposium on High Performance Interconnects (HOTI'05)*, Stanford University, Palo Alto, CA, USA, August 6–8, 2005, pp. 138–143.

[11] S.-M. He, S.-T. Sun, H.-T. Guan, Q. Zheng, Y.-J. Zhao, and W.Gao, "On guaranteed smooth switching for buffered crossbar switches," to appear in *IEEE/ACM Transactions on Networking*.

[12] P. Giaccone, E. Leonardi and D. Shah, "On the maximal throughput of networks with finite buffers and its application to buffered crossbars," *Proceedings IEEE INFOCOM*, 2005.

[13] T. Javidi, R. Magill, and T. Hrabik, "A high-throughput scheduling algorithm for a buffered crossbar switch fabric," in *Proceedings IEEE International Conference on Communications (ICC'01)*, Helsinki, Finland, June 11–14, 2001, pp. 1586–1591.

[14] F. P. Kelly, "Notes on effective bandwidths," *Stochastic Networks: Theory and Applications*, pp. 141–168, Oxford University Press, 1995.

[15] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 1–15, February 1994.

[16] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, pp. 371–381, February 2003.

[17] R. B. Magill, C. E. Rohrs, and R. L. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 606–615, May 2003.

[18] M. Nabeshima, "Performance evaluation of a combined input- and crosspoint-queued switch," *IEICE Transactions on Communications*, vol. E83-B, pp. 737–741, March 2000.

[19] M. J. Neely, E. Modiano, and Y.-S. Cheng, "Logarithmic delay for $N \times N$ packet switches under the crossbar constraint," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 657–668, June 2007.

[20] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: combined input-one-cell-crosspoint buffered switch," in *Proceedings IEEE Workshop on High Performance Switching and Routing (HPSR'01)*, Dallas, TX, USA, May 29–31, 2001, pp. 324–329.

[21] Y. Shen, S. S. Panwar, H. J. Chao, "SQUID: A practical 100% throughput scheduler for crosspoint buffered switches," *Proceedings of IEEE High Performance Switching and Routing Workshop*, 2008.

[22] J. Turner, "Strong performance guarantees for asynchrnonous crossbar schedulers," *Proceedings of IEEE INFOCOM*, 2006.

[23] I. Widjaja and I. Saniee, "Simplified layering and flexible bandwidth with TWIN," in *Proceedings ACM Special Interest Group on Data Communication Conference (SIGCOMM'04)*, Portland, OR, USA, August 30–September 3, 2004, pp. 13–20.

[24] K. Yoshigoe, K. Christensen, and A. Jocob, "The RR/RR CICQ switch: hardware design for 10-Gbps link speed," in *Proceedings IEEE International Conference on Performance, Computing, and Communications (IPCCC'03)*, Phoenix, AZ, USA, April 9–11, 2003, pp. 481–485.

[25] X. Zhang and L. N. Bhuyan, "An efficient algorithm for combined-input-crosspoint-queued (CICQ) switches," in *Proceedings IEEE Global Telecommunications Conference (GLOBECOM'04)*, Dallas, TX, USA, November 29–December 3, 2004, pp. 1168–1173.

[26] X. Zhang, S. R. Mohanty, and L. N. Bhuyan "Adaptive max-min fair scheduling in buffered crossbar switches without speedup," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'07)*, Anchorage, AK, USA, May 6–12, 2007, pp. 454–462.