On Service Guarantees for Input Buffered Crossbar Switches: A Capacity Decomposition Approach by Birkhoff and von Neumann

Cheng-Shang Chang Wen-Jyh Chen Hsiang-Yi Huang
Dept. of Electrical Engineering
National Tsing Hua University
Hsinchu 30043 Taiwan, R.O.C.
Email: cschang@ee.nthu.edu.tw
{wjchen, hyhuang}@hardy.ee.nthu.edu.tw

Abstract— **Based on a decomposition result by Birkhoff and von Neumann for a doubly substochastic matrix, in this paper we propose a scheduling algorithm that is capable of providing service guarantees for input-buffered crossbar switches. Our service guarantees are uniformly good for all non-uniform traffic, and thus imply 100% throughput. The off-line computational complexity to identify the scheduling algorithm is $O(N^{4.5})$ for an $N \times N$ switch. Once the algorithm is identified, its on-line computational complexity is $O(\log N)$ and its on-line memory complexity is $O(N^3 \log N)$. Neither framing nor internal speedup is required for our approach.**

Keywords— **Input buffered switches, crossbar switches, service guarantees, scheduling, generalized processor sharing**

I. Introduction

Memory accessing speed has become a problem of building high speed switches. For a switch with N input ports and N output ports, a shared memory switch is required to read all the N input packets and write them within a packet transmission time (see e.g., Schwartz [25] and Hui [13]). One way to reduce the burden for memory access is to increase the number of parallel lines in the memory bus (e.g., from 32 bits to 64 bits). If the packet size is kept the same, increasing the number of lines in the bus also decreases the time to transmit a packet and thus makes the control of the switch more difficult. The other way is to segregate memory so that the switch can execute read/write operations in parallel. To the extreme, one might allocate a memory block to each input-output pair. In this case, it requires N^2 segregated memory blocks. Such memory complexity may not be scalable for switches with a large number of input ports and output ports. Also, as memory cannot be shared, it is expected that packet losses due to buffer overflows will be much larger than those in shared memory switches.

This research is supported in part by the National Science Council, Taiwan, R.O.C., under Contracts NSC-88-2213-E007-004 and NSC-87-2219-E007-005.

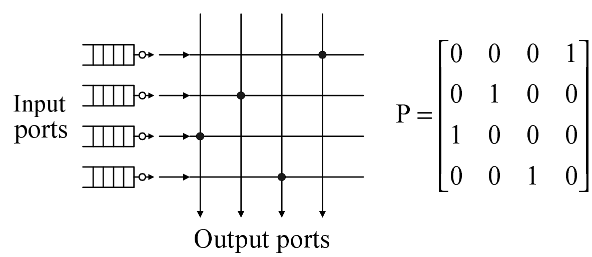

Fig. 1. A 4×4 input-buffered crossbar switch

Crossbar switches, as a compromise of all these, have received a lot of attention recently (see e.g., [26], [19], [17], [8], [20], [7], [15], [27], [14]). An input buffered crossbar switch with N input ports and N output ports has a segregated buffer for each input port. In such a switch, time is slotted and synchronized so that packets from different input buffers can be read out simultaneously within a time slot. In a time slot, a crossbar switch sets up a connection pattern corresponding to a permutation matrix (see Figure 1). As a permutation matrix is a one-to-one mapping from input ports to output ports, packets destined to the same output ports cannot be transmitted at the same time. As discussed in [20], such limitation causes two potential problems: low throughput due to head-of-line (HOL) blocking and the difficulty of controlling packet delay. By allowing the switch to transmit non-HOL packets, it is shown in [19], [20] that 100% throughput can be achieved by certain weighted matching algorithms. To control packet delays, it is shown in [8], [27], [7], [15] that certain deterministic guarantees, such as queue length and delay, can be obtained by internal speedup. In particular, a speedup of 2 is sufficient to mimic the behavior of an output-buffered switch (see [8], [27]).

In the recent paper [14], Hung, Kesidis and McKeown used an idling weighted round robin (WRR) algorithm in [2] to achieve rate guarantee for each input-output pair without internal speedup. This in turn implies 100% throughput (if the rate for each input-output pair is known). A similar approach is also addressed by Lee and Lam [17]. As the usual WRR algorithm, such an approach requires to choose a frame size. A large frame size implies a large worst case packet delay and a large memory requirement for the storage of all the connection patterns in a frame. On the other hand, a small frame size implies a large rate granularity (the minimum rate allocated to an input-output pair). As a result, the WRR algorithm fails to provide uniform service guarantees for all non-uniform traffic.

In this paper, we propose an algorithm that is capable of providing uniform service guarantees for all non-uniform traffic. To be precise, let $r_{i,j}$ be the rate assigned to the traffic from input i to output j for an $N \times N$ input-buffered crossbar switch, and $C_{i,j}(t)$ be the cumulative number of slots that are assigned to the traffic from input i to output j by time t. We show that if

$$\sum_{i=1}^{N} r_{i,j} \le 1, \quad \forall j, \quad \text{and} \tag{1}$$

$$\sum_{j=1}^{N} r_{i,j} \le 1, \quad \forall i, \tag{2}$$

then there is a scheduling algorithm such that

$$C_{i,j}(t) - C_{i,j}(s) \ge r_{i,j}(t-s) - s_{i,j}, \tag{3}$$

for all i, j, $s \le t$, and some $s_{i,j} \le N^2 - 2N + 2$. The two conditions in (1) and (2), called "no overbooking" conditions in [14], are intuitively clear as they simply state that neither the total rate to an output port nor the total rate coming out from an input port can be larger than 1. The computational complexity to identify a scheduling algorithm to guarantee (3) is $O(N^{4.5})$ for an $N \times N$ input buffered crossbar switch. Once the algorithm is identified, its on-line computational complexity is $O(\log N)$ and its on-line memory complexity is $O(N^3 \log N)$.

Our approach eliminates the problem of choosing the frame size in the WRR algorithm. This is done via a decomposition approach by Birkhoff [4] and von Neumann [28]. The decomposition approach reduces the two-dimensional rate assignment problem to a one-dimensional problem so that the Packetized Generalized Processor Sharing (PGPS) algorithm in Parekh and Gallager [23] (the Weighted Fair Queueing (WFQ) in Demers, Keshav, and Shenkar [11]) can be applied. As in [14], our approach also achieves 100% throughput and guarantees the rate for each input-output pair without internal speedup.

The paper is organized as follows: in Section II, we illustrate how one applies the Birkhoff-von Neumann decomposition. In Section III, we present the on-line algorithm and prove its service guarantee property. We conclude the paper in Section IV by discussing possible applications of our algorithm to achieve deterministic guarantees, 100 % throughput, stochastic guarantees, and differentiated services.

II. Preliminaries on doubly stochastic matrices

A nonnegative matrix $R = (r_{i,j})$ that satisfies the conditions in (1) and (2) is known to be *doubly substochastic* (see e.g., Marshall and Olkin [18]). For such a matrix, we will illustrate how one uses the decomposition approach by Birkhoff [4] and von Neumann [28] to find a set of positive numbers ϕ_k and permutation matrices P_k, $k = 1, \ldots, K$ for some $K \le N^2 - 2N + 2$ that satisfies

$$R \le \sum_{k=1}^{K} \phi_k P_k, \quad \text{and} \tag{4}$$

$$\sum_{k=1}^{K} \phi_k = 1. \tag{5}$$

A nonnegative matrix $R = (r_{i,j})$ is called *doubly stochastic* if both inequalities in (1) and (2) are equalities. Clearly, one can reduce the values of the elements in a doubly stochastic matrix to obtain a doubly substochastic matrix. It turns out there is a converse to this as presented in [18].

Proposition 1: **(von Neumann [28])** If a matrix $R = (r_{i,j})$ is doubly substochastic, then there exists a doubly stochastic matrix $\tilde{R} = (\tilde{r}_{i,j})$ such that

$$r_{i,j} \leq \tilde{r}_{i,j}, \quad \forall i, j.$$

This can be constructed by the following algorithm.

Algorithm 1:
(i) If the sum of all the elements in R is less than N, then there exists an element (i, j) such that $\sum_n r_{i,n} < 1$ and $\sum_m r_{m,j} < 1$.
(ii) Let $\epsilon = 1 - \max[\sum_n r_{i,n}, \sum_m r_{m,j}]$. Add ϵ to the element in the $(i, j)^{th}$ position to obtain a new matrix R_1. Then in R_1, the number of row sums and column sums that are strictly smaller than 1 is at least one less than that in R.
(iii) Repeat this procedure until the sum of all the elements is equal to N. In this case, we derive a doubly stochastic matrix.

Clearly, the above algorithm will be ended after at most $2N - 1$ steps as the last step will make one row sum and one column sum to 1. As the matrix is $N \times N$, the computational complexity of finding an element (i, j) with $\sum_n r_{i,n} < 1$ and $\sum_m r_{m,j} < 1$ is $O(N^2)$. Thus, the computational complexity of Algorithm 1 is $O(N^3)$.

We note that if $r_{i,j}$'s satisfy (1) and (2) with *strict* inequalities, then there exists a doubly stochastic matrix $\tilde{R} = (\tilde{r}_{i,j})$ such that

$$r_{i,j} < \tilde{r}_{i,j}, \quad \forall i, j. \tag{6}$$

To see this, let

$$r_{max} = \max\left[\max_i[\sum_n r_{i,n}], \max_j[\sum_m r_{m,j}]\right]$$

be the maximum among all the row sums and column sums. As the inequalities in (1) and (2) are strict, $r_{max} < 1$. Construct a matrix $R' = \frac{1}{r_{max}} R$. Then $R' = (r'_{i,j})$ is still doubly substochastic and we can use Algorithm 1 to construct a doubly stochastic matrix $\tilde{R} = (\tilde{r}_{i,j})$ such that for all i, j,

$$\tilde{r}_{i,j} \geq r'_{i,j} = \frac{1}{r_{max}} r_{i,j} > r_{i,j}. \tag{7}$$

One of the most important results for doubly stochastic matrices is that the convex hull of the permutation matrices coincides with the set of doubly stochastic matrices.

Proposition 2: **(Birkhoff [4])** For a doubly stochastic matrix $\tilde{R}$, there exists a set of positive numbers ϕ_k and permutation matrices P_k such that

$$\tilde{R} = \sum_k \phi_k P_k.$$

Let e be the column vector with all its elements being 1. As $\tilde{R}$ is doubly stochastic,

$$e = \tilde{R}e = \sum_k \phi_k (P_k e) = (\sum_k \phi_k)e.$$

This shows that

$$\sum_k \phi_k = 1. \tag{8}$$

In the following, we present an algorithm to obtain the set of positive numbers ϕ_k and permutation matrices P_k. As indicated in [18], the algorithm was originated by Dulmage and Halperin [12].

Algorithm 2:
(i) For a doubly stochastic matrix $\tilde{R}$, let $(i_1, i_2, \ldots, i_N)$ be a permutation of $(1, 2, \ldots, N)$ such that $\prod_{k=1}^N \tilde{r}_{k,i_k} > 0$. The existence of such a permutation for a doubly stochastic matrix is given by Mirsky [21], pp. 185 and Berge [3], pp. 105.
(ii) Let P_1 be the permutation matrix corresponding to $(i_1, i_2, \ldots, i_N)$, and $\phi_1 = \min_{1 \leq k \leq N}[r_{k,i_k}]$. Define the matrix R_1 by

$$R_1 = \tilde{R} - \phi_1 P_1.$$

(iii) If $\phi_1 = 1$, then

$$R_1 e = \tilde{R}e - P_1 e = \mathbf{0},$$

where $\mathbf{0}$ is the column vector with all its elements being zero. In this case, the matrix R_1 is the zero matrix as all the column sums are zero. Thus, we have completed the representation.
(iv) If $\phi_1 < 1$, then the matrix $\frac{1}{1-\phi_1} R_1$ is doubly stochastic and we can continue the decomposition from (i).

As each iteration reduces at least one element to zero, it is easy to see that the algorithm will be stopped by at most $N^2 - N + 1$ steps (the last one will reduce N elements to zero). In fact, as shown in [18] and references therein, the algorithm will be stopped by at most $N^2 - 2N + 2$ steps. The number $N^2 - 2N + 2$ cannot be reduced by a smaller number for the worse case. The main computation complexity of Algorithm 2 is to find a permutation $(i_1, i_2, \ldots, i_N)$ such that $\prod_{k=1}^N \tilde{r}_{k,i_k} > 0$. This can be done by converting the matrix $\tilde{R}$ into a $0-1$ valued adjacency matrix of a bipartite graph and finding a maximum matching in that bipartite graph (in this case, a maximum matching is ensured to be perfect). It is known (see e.g., [22], Theorem 10.2) that the computational complexity for the bipartite matching problem is $O(N^3)$ via the alternating path algorithm, and can be improved to $O(N^{2.5})$ via mapping the problem to the maximum flow problem. Thus, the computational complexity of Algorithm 2 is $O(N^{4.5})$.

We note that if there is an integer f such that the matrix $f\tilde{R}$ contains all integer-valued elements, then Algorithm 2 will be stopped by at most f steps. To see this, observe that

$$\frac{1}{1-\phi_1}R_1 = \frac{\tilde{R}-\phi_1 P_1}{1-\phi_1} = \frac{f\tilde{R}-f\phi_1 P_1}{f-f\phi_1}.$$

For $\phi_1 < 1$, the matrix $(f-f\phi_1)(\frac{1}{1-\phi_1}R_1)$ is a matrix with all integer-valued elements. Since $f\phi_1$ is an integer not less than 1, we only need to multiply the matrix after one step by a smaller number than f for it to become a matrix with all integer-valued elements. Thus, Algorithm 2 will be stopped by at most f steps. This shows that the WRR algorithm in [14], [17] is a special case of Algorithm 2 when the elements in the matrix $\tilde{R}$ are rational numbers. Besides, Algorithm 2 generates a more compact representation than the Slepian-Duguid algorithm used in the WRR algorithm.

III. Algorithm description and analysis

In this section, we present our on-line algorithm and its analysis. From the constructions in Algorithms 1 and 2, we can find a set of positive numbers ϕ_k and permutation matrices P_k, $k=1,\ldots,K$ for some $K \le N^2-2N+2$ that satisfies

$$R \le \sum_{k=1}^K \phi_k P_k, \quad \text{and} \tag{9}$$

$$\sum_{k=1}^K \phi_k = 1. \tag{10}$$

In view of (9) and (10), the amount of time that the crossbar switch sets its connection pattern according to the permutation matrix P_k should be proportional to ϕ_k. As time is slotted, we can use the Packetized Generalized Processor Sharing (PGPS) algorithm in Parekh and Gallager [23] (or the Weighted Fair Queueing (WFQ) in Demers, Keshav, and Shenkar [11]) to approximate the scheme.

Algorithm 3:
(i) Assign a class of tokens for each permutation matrix P_k, $k=1,\ldots,K$.
(ii) At time 0, generate a token for each class. Assign the virtual finishing time of the first class k token by

$$F_k^1 = \frac{1}{\phi_k}.$$

Sort these K tokens in an increasing order of the virtual finishing times.
(iii) The switch serves the tokens in an increasing order of the virtual finishing times. Once a class k token is served in a time slot, the switch sets its connection pattern according to the permutation matrix P_k for that time slot.
(iv) After the $(\ell-1)^{th}$ class k token is served, the switch generates the ℓ^{th} class k token and assign the virtual finishing time by

$$F_k^\ell = F_k^{\ell-1} + \frac{1}{\phi_k}.$$

Insert the newly generated token in the sorted token list and repeat the algorithm from (iii).

Note that Algorithm 3 is in fact a special case of PGPS. It corresponds to the case that there is an infinite number of packets at time 0 for PGPS. As the algorithm generates a new token of the same class after a token is served, there are exactly K classes of tokens in the sorted token list. Since the tokens are always backlogged, there is no need to keep track of the virtual time in [23]. Thus, Algorithm 3 is independent of the buffer contents at any time.

The complexity of Algorithm 3 is to insert every new token into the sorted token list. As there are K tokens in the sorted list and $K \le N^2-2N+2$, the computational complexity of Algorithm 3 is $O(\log N)$. Since it takes $N\log_2 N$ bits to describe a permutation matrix and there are $O(N^2)$ permutation matrices needed to be stored in Algorithm 3, the memory complexity of Algorithm 3 is $O(N^3 \log N)$.

Now we present the main result of our paper. Let $C_{i,j}(t)$ be the cumulative number of slots that are assigned to the traffic from input i to output j by time t. Also, let $p_{k,i,j}$ be the element at the (i,j) position of the permutation matrix P_k obtained from Algorithm 2. Note that $p_{k,i,j}$ only takes values 0 and 1.

Define $E_{i,j} = \{k : p_{k,i,j} = 1\}$ to be the subset of $\{1, 2, \dots, K\}$ such that for all k in $E_{i,j}$, the permutation matrix P_k has a nonzero element at the (i, j) position.

Theorem 3: For any rate matrix $R = (r_{i,j})$ that satisfies (1) and (2), Algorithms 1,2 and 3 generates a scheduling policy that guarantees

$$\sum_{k\in E_{i,j}} \phi_k(t-s) - s_{i,j} \le C_{i,j}(t) - C_{i,j}(s) \le \sum_{k\in E_{i,j}} \phi_k(t-s) + s_{i,j}, \tag{11}$$

where

$$s_{i,j} = \min[K, |E_{i,j}| + \sum_{k\in E_{i,j}} \phi_k(K-1)], \tag{12}$$

and $|E_{i,j}|$ is the number of elements in the set $E_{i,j}$.

Note that K and ϕ_k's are the parameters obtained from Algorithm 2. As $K \le N^2-2N+2$ in Algorithm 2, we have $s_{i,j} \le N^2 - 2N + 2$. Also, we have from (9) that

$$\sum_{k\in E_{i,j}} \phi_k = \sum_{k=1}^{K} \phi_k p_{k,i,j} \ge r_{i,j}. \tag{13}$$

This in turn implies

$$C_{i,j}(t) - C_{i,j}(s) \ge r_{i,j}(t-s) - N^2 + 2N - 2,$$

for all i, j, and $s \le t$.

For example, consider the following rate matrix

$$R = \begin{bmatrix} 0.10 & 0 & 0.13 & 0.40 \\ 0.11 & 0.07 & 0.60 & 0.05 \\ 0 & 0.32 & 0.12 & 0.23 \\ 0.51 & 0.23 & 0.04 & 0.12 \end{bmatrix}. \tag{14}$$

Using Algorithm 1 with the sequence of pivots $(1,1), (1,3), (2,2), (3,2), (3,3), (3,4), (4,4)$, we have the following double stochastic matrix

$$\tilde{R} = \begin{bmatrix} 0.38 & 0 & 0.22 & 0.40 \\ 0.11 & 0.24 & 0.60 & 0.05 \\ 0 & 0.53 & 0.14 & 0.33 \\ 0.51 & 0.23 & 0.04 & 0.22 \end{bmatrix}. \tag{15}$$

From Algorithm 2, we have the following decomposition

$$\begin{aligned}\tilde{R} = 0.14 &\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{bmatrix} + 0.04 \begin{bmatrix} 1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0 \end{bmatrix} \\ + 0.40 &\begin{bmatrix} 0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0 \end{bmatrix} + 0.08 \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{bmatrix} \\ + 0.01 &\begin{bmatrix} 0&0&1&0\\0&0&0&1\\0&1&0&0\\1&0&0&0 \end{bmatrix} + 0.12 \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0 \end{bmatrix} \\ + 0.11 &\begin{bmatrix} 0&0&1&0\\1&0&0&0\\0&0&0&1\\0&1&0&0 \end{bmatrix} + 0.10 \begin{bmatrix} 0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0 \end{bmatrix}.\end{aligned}$$

Using (12), we can find $S = (s_{i,j})$ as follows:

$$S = \begin{bmatrix} 6.66 & 0 & 4.54 & 3.8 \\ 1.77 & 3.68 & 7.2 & 2.35 \\ 0 & 7.71 & 1.98 & 5.31 \\ 6.57 & 3.61 & 1.28 & 3.54 \end{bmatrix}. \tag{16}$$

The rest of the section is devoted to the proof of Theorem 3. Let τ_k^ℓ be the index of the time slot that the ℓ^{th} class k token is served, and $D_k(t)$ be the cumulative number of class k tokens served by time t. These two quantities are related via the following inversion formula

$$D_k(t) = \sup\{\ell : \tau_k^\ell \le t\}. \tag{17}$$

Lemma 4: For all $k = 1, \dots, K$,

$$\lfloor \phi_k t \rfloor \le D_k(t) \le \lfloor \phi_k(t+K-1) \rfloor, \tag{18}$$

where $\lfloor x \rfloor$ is the largest integer that is less than or equal to x.

Using the inequality $x - 1 < \lfloor x \rfloor \le x$, one has

$$\phi_k t - 1 < D_k(t) \le \phi_k(t+K-1). \tag{19}$$

Proof. As Algorithm 3 serves the tokens in an increasing order of the virtual finishing times, in the worst case the ℓ^{th} class k token is served after all the tokens with virtual finishing times not greater than F_k^ℓ are served. Note that

$$F_k^\ell = \ell/\phi_k.$$

The total number of tokens that have virtual finishing times not greater than F_k^ℓ is

$$\ell + \sum_{j \neq k} \lfloor (\ell \phi_j)/\phi_k \rfloor. \tag{20}$$

As the algorithm serves a token per time slot, we have

$$\begin{aligned} \tau_k^\ell &\leq \ell + \sum_{j \neq k} \lfloor (\ell \phi_j)/\phi_k \rfloor \\ &\leq \ell + \sum_{j \neq k} (\ell \phi_j)/\phi_k = \ell/\phi_k = F_k^\ell. \end{aligned}$$

Thus, every token is served not later than its virtual finishing time. From the inversion formula in (17), we then have

$$D_k(t) \geq \sup\{\ell : F_k^\ell \leq t\} = \lfloor \phi_k t \rfloor.$$

We prove the second inequality in (18) by contradiction. Suppose that for some k and some t,

$$D_k(t) > \lfloor \phi_k (t + K - 1) \rfloor.$$

As $D_k(t)$ is an integer, this is equivalent to

$$D_k(t) \geq \lfloor \phi_k (t + K - 1) \rfloor + 1.$$

From the inversion formula in (17), the $\lfloor \phi_k (t + K - 1) \rfloor + 1^{th}$ class k token must be served not later than t, i.e.,

$$\tau_k^{\lfloor \phi_k (t+K-1) \rfloor + 1} \leq t.$$

As the algorithm serves the tokens in an increasing order of their virtual finishing times, those tokens with virtual finishing times less than $F_k^{\lfloor \phi_k (t+K-1) \rfloor + 1}$ must also be served not later than t. From the strict inequality $\lfloor x \rfloor + 1 > x$ and (20), it follows that the total number of tokens served not later than t is strictly larger than

$$\phi_k (t + K - 1) + \sum_{j \neq k} \lfloor (\phi_k (t + K - 1) \phi_j)/\phi_k \rfloor.$$

Since there are exact t tokens served by time t, we have

$$\begin{aligned} t &> \phi_k (t + K - 1) + \sum_{j \neq k} \lfloor (\phi_k (t + K - 1) \phi_j)/\phi_k \rfloor \\ &> \phi_k (t + K - 1) + \sum_{j \neq k} ((t + K - 1) \phi_j - 1) \\ &= t, \end{aligned}$$

and we reach a contradiction. ■

Proof. (Theorem 3) We first prove the lower bound in (11). Note that

$$C_{i,j}(t) = \sum_{k \in E_{i,j}} D_k(t).$$

Thus, it suffices to show that

$$\begin{aligned} &\sum_{k \in E_{i,j}} D_k(t) - \sum_{k \in E_{i,j}} D_k(s) \\ &\geq (\sum_{k \in E_{i,j}} \phi_k)(t - s) - K, \end{aligned} \tag{21}$$

and that

$$\begin{aligned} &\sum_{k \in E_{i,j}} D_k(t) - \sum_{k \in E_{i,j}} D_k(s) \\ &\geq (\sum_{k \in E_{i,j}} \phi_k)(t - s) - |E_{i,j}| - \sum_{k \in E_{i,j}} \phi_k (K - 1). \end{aligned} \tag{22}$$

We now show (21). As exact one token is served in a time slot,

$$\sum_{k=1}^{K} D_k(t) = t, \quad \forall t. \tag{23}$$

Thus, we have from (23) and the lower bound in (19) that

$$\begin{aligned} \sum_{k \in E_{i,j}} D_k(t) &= t - \sum_{k \notin E_{i,j}} D_k(t) \\ &\leq \sum_{k \in E_{i,j}} \phi_k t + (K - |E_{i,j}|), \quad \forall t. \end{aligned} \tag{24}$$

Using the lower bound in (19) and (24) yields

$$\begin{aligned} &\sum_{k \in E_{i,j}} D_k(t) - \sum_{k \in E_{i,j}} D_k(s) \\ &\geq \sum_{k \in E_{i,j}} (\phi_k t - 1) - \sum_{k \in E_{i,j}} \phi_k s - (K - |E_{i,j}|) \\ &= \sum_{k \in E_{i,j}} \phi_k (t - s) - K. \end{aligned}$$

To see (22), note from the lower bound and the upper bound in (19) that

$$\begin{aligned} &\sum_{k \in E_{i,j}} D_k(t) - \sum_{k \in E_{i,j}} D_k(s) \\ &\geq \sum_{k \in E_{i,j}} (\phi_k t - 1) - \sum_{k \in E_{i,j}} (\phi_k (s + K - 1)) \\ &= \sum_{k \in E_{i,j}} \phi_k (t - s) - |E_{i,j}| - \sum_{k \in E_{i,j}} \phi_k (K - 1). \end{aligned}$$

To prove the upper bound in (11), we simply interchange the upper bound and the lower bound used for s and t in the proof for the lower bound in (11). ■

IV. DISCUSSIONS

In this paper, we proposed a scheduling algorithm to achieve service guarantees for an input-buffered crossbar switch. Such service guarantees, in conjunction with proper input traffic assumptions, can be used to obtain deterministic guarantees, 100% throughput, and stochastic guarantees. We will also show how one modifies our algorithm to provide differentiated services.

Deterministic guarantees:

Recently, a set of papers (e.g., [27], [15], [7]) have shown that deterministic guarantees, such as delay and queue length, can be achieved for an input buffered crossbar switch by internal speedup. The result in Theorem 3 shows that there is no need for internal speedup to achieve deterministic guarantees. To be precise, assume that the traffic $A_{i,j}$ from the i^{th} input port to the j^{th} output port is $(\sigma_{i,j}, r_{i,j})$-upper constrained, i.e.,

$$A_{i,j}(t) - A_{i,j}(s) \leq r_{i,j}(t-s) + \sigma_{i,j}, \qquad (25)$$

where $A_{i,j}(t)$ is the cumulative number of packets that are destined to the j^{th} output port from the i^{th} input port by time t. If $r_{i,j}$'s satisfy the rate conditions in (1) and (2), then one can apply the calculus in [9], [10] and Theorem 3 to show that the buffer size needed for $A_{i,j}$ is at most $\sigma_{i,j} + s_{i,j}$, and that the maximum delay from the i^{th} input port to the j^{th} port is bounded above by $\lceil (s_{i,j} + \sigma_{i,j})/r_{i,j} \rceil$ (assuming FIFO for the traffic $A_{i,j}$), where $\lceil x \rceil$ is the smallest integer that is not less than x. We note that our guarantee in Theorem 3 is a very strong one and it implies the usual service curve guarantee in [24], [6], [1], [16]. Such a strong guarantee allows us to provide per flow QoS for flows aggregated within $A_{i,j}$ via the Service Curve based Earliest Deadline (SCED) policy in [24], [10].

100% throughput:

Suppose that the traffic $A_{i,j}$ is now a stationary and ergodic stochastic process with the mean rate $r_{i,j}$, $i, j = 1, \ldots, N$. If $r_{i,j}$'s satisfy the conditions in (1) and (2) with strict inequalities, then it is known (see e.g., [19], [20]) that 100% throughput can be achieved for input buffered crossbar switch by using some weighted matching scheduling algorithm with computational complexity $O(N^{2.5})$. If $r_{i,j}$'s are known to the switch, then our algorithm (along with the additional step in (7)) reduces the on-line computational complexity to $O(\log N)$. In comparison with the algorithms in [19], [20], theirs is "greedy at all time" and ours is "good on average." As perfect matching is not likely for all time, it is quite intuitive why a "good on average" policy like ours also achieves 100% throughput.

In the case that the mean rates are not available or the stochastic processes are not stationary, on-line measurement and adaptation might be needed for our algorithm. We have performed some preliminary experiments using a simple recursive estimator. Our experimental results show that 100% throughput is still achievable for Bernoulli routing. These results will be reported separately.

Stochastic guarantees:

Suppose that for all i, j, the traffic $A_{i,j}$ is $(\sigma_{i,j}(\theta), r_{i,j}(\theta))$-upper constrained for some $\theta > 0$, i.e.,

$$\begin{aligned} &\frac{1}{\theta} \log \mathsf{E} e^{\theta(A_{i,j}(t) - A_{i,j}(s))} \\ &\leq r_{i,j}(\theta)(t-s) + \sigma_{i,j}(\theta), \quad \forall s \leq t. \end{aligned} \qquad (26)$$

As in (1) and (2), assume that

$$\begin{aligned} \sum_i r_{i,j}(\theta) < 1, \quad \forall j \text{ and} \\ \sum_j r_{i,j}(\theta) < 1, \quad \forall i. \end{aligned}$$

Let $q_{i,j}(t)$ be the number of packets from $A_{i,j}$ that are waiting for transmission at time t. Applying the calculus in [5], one can easily show that for all t and x

$$\mathsf{P}(q_{i,j}(t) \geq x) \leq \beta(\theta) e^{-\theta x},$$

for some finite constant $\beta(\theta)$.

Differentiated services:

Continue the discussion from the part for deterministic guarantees. In addition to the traffic with guaranteed rates, there is best-effort traffic. Suppose that $r_{i,j}$'s satisfy the rate conditions in (1) and (2) with strict inequalities. Then we can modify our algorithm so that the best-effort traffic can use the remaining bandwidth left from serving the traffic with guaranteed rates.

Using the construction in (7), we can derive a doubly stochastic matrix $\tilde{R} = (\tilde{r}_{i,j})$ such that

$$\tilde{r}_{i,j} \geq \frac{1}{r_{max}} r_{i,j}, \tag{27}$$

where r_{max} is the maximum of the row sums and column sums. From Algorithm 2, we can find ϕ_k and P_k such that

$$\tilde{R} = \sum_{k=1}^{K} \phi_k P_k.$$

Thus, we have from (27) that

$$R \leq \sum_{k=1}^{K} (r_{max}\phi_k) P_k. \tag{28}$$

Let $\tilde{\phi}_k = r_{max}\phi_k$, $k = 1, \ldots, K$, $\tilde{\phi}_{K+1} = (1 - r_{max})$, and P_{K+1} be the matrix with all its elements being 0. Rewrite (28) as follows:

$$R \leq \sum_{k=1}^{K+1} \tilde{\phi}_k P_k. \tag{29}$$

Now we can use (29) to run Algorithm 3. The only difference is when a class $K + 1$ token is served. When this happens, the switch may set up any connection pattern to serve best-effort traffic (the connection pattern could be decided by some matching algorithm). Clearly, Theorem 3 still holds with K being replaced by $K + 1$. Thus, the modified algorithm not only achieves rate guarantees, but also allows the best-effort traffic to use the remaining bandwidth.

References

[1] R. Agrawal and R. Rajan, "Performance bounds for guaranteed and adaptive services," *IBM RC 20649*, 1996. Also in *Proc. 34th Allerton Conf. on Comm., Cont. & Comp.*, Monticello, IL, Oct. 1996.

[2] T. Anderson, S. Owicki, J. Saxes and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. on Computer Systems*, Vol. 11, pp. 319-352, 1993.

[3] C. Berge, *The Theory of Graphs and Its Applications* (translated by A. Doig). New York: Wiely, 1962.

[4] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, Vol. 5, pp. 147-151, 1946.

[5] C.S. Chang, "Stability, queue length and delay of deterministic and stochastic queueing networks," *IEEE Transactions on Automatic Control*, Vol.39, pp. 913-931, 1994.

[6] C.S. Chang, "On deterministic traffic regulation and service guarantees: a systematic approach by filtering," *IEEE Transactions on Information Theory*, Vol. 44, pp. 1097-1110, 1998.

[7] A. Charny, P. Krishna, N. Patel and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup," *IEEE IWQoS'98*, pp. 235-244, Napa, California, 1998.

[8] S.-T. Chuang, A. Goel, N. McKeown and B. Prabhkar, "Matching output queueing with a combined input output queued switch," *Computer Systems Technical Report CSL-TR-97-738*, 1997. Available from *http://tiny-tera.stanford.edu/~nickm/papers.html*.

[9] R.L. Cruz, "A calculus for network delay, Part I: Network elements in isolation," *IEEE Transactions on Information Theory*, Vol. 37, pp. 114-131, 1991.

[10] R.L. Cruz, *Lecture Notes on Quality of Service Guarantees*, 1998.

[11] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm," in *Proc. SIGCOMM'89*, pp. 1-12, Austin, TX, Sept. 1989.

[12] L. Dulmage and I. Halperin, "On a theorem of Frobenius-König and J. von Neumann's game of hide and seek, *Trans. Roy. Soc. Canada III(3)*, Vol. 49, pp. 23-29, 1955.

[13] J. Hui, *Switching and Traffic Theory for Integrated Broadband Networks.* Boston: Kluwer Academic Publishers, 1990.

[14] A. Hung, G. Kesidis and N. Mckeown, "ATM input-buffered switches with guaranteed-rate property," *Proc. IEEE ISCC'98*, Athens, pp. 331-335, 1998.

[15] P. Krishna, N.S. Patel, A. Charny and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE IWQoS'98*, pp. 225-234, Napa, California, 1998.

[16] J.Y. Le Boudec, " Application of network calculus to guaranteed service networks," *IEEE Transactions on Information Theory*, Vol. 44, pp. 1087-1096, 1998.

[17] T.T. Lee and C.H. Lam, "Path switching-a quasi-static routing scheme for large scale ATM packet switches," *IEEE Journal on Selected Areas of Communications*, Vol. 15, pp. 914-924, 1997.

[18] A.W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*. New York: Academic Press, 1979.

[19] N. McKeown, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Proc. IEEE INFOCOM'96*, pp. 296-302, 1996.

[20] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *Proc. IEEE INFOCOM'98.*

[21] L. Mirsty, *Transversal Theory.* New York: Academic Press, 1971.

[22] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithm and Complexity.* New Jersey: Prentice-Hall, 1982.

[23] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the single-node case," *IEEE/ACM Transactions on Networking*, Vol. 1, pp. 344-357, 1993.

[24] H. Sariowan, R.L. Cruz and G.C. Polyzos, "SCED: a generalized scheduling policy for guaranteeing quality-of-service," submitted to *IEEE/ACM Transactions on Networking*.

[25] M. Schwartz, *Broadband Integrated Networks*. New Jersey: Prentice Hall, 1996.

[26] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *Proc. IEEE INFOCOM'95*, pp. 960-968, 1995.

[27] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," *IEEE IWQoS'98*, pp. 218-224, Napa, California, 1998.

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