Input-Queued Switching with QoS Guarantees

Shizhao Li
Department of Electrical & Computer Engineering
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, U.S.A.
sxl3756@megahertz.njit.edu

Nirwan Ansari
Department of Information Engineering
The Chinese University of Hong Kong
Sha Tin, N.T., Hong Kong
nansari@ie.cuhk.edu.hk
(on leave from NJIT)

Abstract—**Input-queued switching architecture is becoming an attractive alternative for designing very high speed switches owing to its scalability. Tremendous efforts have been made to overcome the throughput problem caused by contentions occurred at the input and output sides of the switches. However, no QoS guarantees can be provided by the current input-queued switch design.**

In this paper, a frame based scheduling algorithm, referred to as Store-Sort-and-Forward (SSF), is proposed to provide QoS guarantees for input-queued switches without requiring speedup. SSF uses a framing strategy in which the time axis is divided into constant-length frames, each made up of an integer multiple of time slots. Cells arrived during a frame are first held in the input buffers, and are then "sorted-and-transmitted" within the next frame. A bandwidth allocation strategy and a cell admission policy are adopted to regulate the traffic to conform to the (r, T) traffic model. A strict sense 100% throughput is proved to be achievable by rearranging the cell transmission orders in each input buffer, and a sorting algorithm is proposed to order the cell transmission. The SSF algorithm guarantees bounded end-to-end delay and delay jitter. It is proved that a perfect matching can be achieved within $N(\ln N + O(1))$ effective moves.

I. Introduction

Scheduling is one of the most important issues in providing guaranteed quality of service (QoS) in packet-switching networks. Scheduling algorithms which have been studied in the literature can be classified into three categories according to the adopted switching architecture: input scheduling, output scheduling, and combined input output scheduling.

Most of the early studies focused on the output scheduling owing to its conceptual simplicity. By assuming that cells are readily available to be transmitted to the output links upon entering a switch, many proposed algorithms are able to provide QoS guarantees (for an overview, see [1]). However, the output queueing architecture suffers from the scalability problem. Since more than one cell can arrive at the switch in a given time slot heading for the same output, the fabric and output buffers should have the capability to accommodate all of the cells to avoid cell loss and to meet certain delay bounds. In the worst case, an $N \times N$ switch has to run N times faster than a single link when all N inputs receive cells directed to the same output in a time slot. Thus, the buffers, switch fabric, and control system have to be sped up proportionally to the number of input or output links, thus severely limiting the switch capacity.

This work has been supported in part by an AT&T Foundation special purpose grant.

The input queueing architecture, on the other hand, eliminates the scalability problem. Since buffers are placed at the input side of the switch, the fabric and buffers can run at the same speed as a single link without causing cell loss. Owing to its scalability, input queueing is receiving attention in both the research and commercial communities [2], [3], [4], [5], [6]. The major problem with the input queueing architecture is its low throughput. It was shown [7] that the throughput of an input-queued switch is limited to 58.6% of its capacity for independent Bernoulli traffic if a single FIFO queue is used in each input buffer. This is mainly owing to the head of line (HOL) blocking, i.e., if cells at the head of the FIFO queues are blocked, cells stored behind them cannot be transmitted even if their destination outputs are open. Since the publication of [7], many works have indicated that the throughput of an input-queued switch can be improved by adopting a well designed buffering scheme [8], [9]. HOL blocking can be eliminated by adopting virtual output queueing, in which each input maintains a separate FIFO queue for each output. Cells at the heads of the FIFO queues are all accessible to the scheduler. Since the head cells are directed to different outputs, even if one cell is blocked, the others are still available for transmission. However, since only one cell can be transmitted from each input or to each output in any given time slot, contentions occur at both the input and output sides, thus still limiting the throughput of an input-queued switch. Most studies on input scheduling focus on improving the throughput rather than providing deterministic QoS guarantees. Maximizing the throughput is equivalent to the matching problem in a bipartite graph [10]. An iterative algorithm called iSLIP, which is a maximum size matching based scheduler, can achieve asymptotically 100% throughput for uniform traffic[1] [5]. Maximum weight matching based algorithms have been proposed later to achieve 100% throughput for non-uniform traffic [4], [6]. However, none of the existing input scheduling algorithms has been able to provide deterministic QoS guarantees.

There has been a trade off between QoS guarantees and scalability: the input queueing architecture is scalable but cannot provide guaranteed QoS, while the output queueing architecture can provide guaranteed QoS but is not scalable. Lately, there is a trend to adopt combined input output queue-

[1] All arrivals have the same rate, and are directed equally to all outputs.

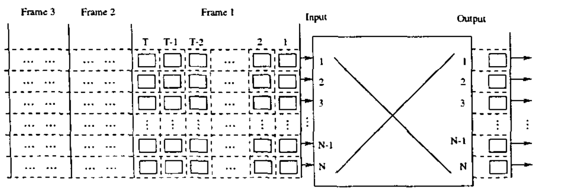

Fig. 1. An $N \times N$ input-queued switch with time axis divided into frames

ing (CIOQ), in which buffers are placed at both the input and output sides of a switch [11], [12]. It has been proven in [11] that a speedup of 2 is sufficient for a CIOQ switch to behave identically to an output-queued switch which employs work-conserving and monotonic scheduling discipline.

With the same buffer access and fabric speed, a switch which does not require speedup can provide N times the capacity of a switch which requires a speedup of N. Thus, speedup should be kept as low as possible, and can only be eliminated by using solely input queueing. In this paper, an input scheduling algorithm, referred to as Store-Sort-and-Forward (SSF), is proposed and proved to be able to provide guaranteed end-to-end delay and delay jitter bounds, and to achieve strict sense 100% throughput with no speedup. As opposed to existing input scheduling algorithms [4], [5], [6], [13] which employ the work-conserving discipline, SSF uses a framing strategy, which is non-work-conserving. A switch implementing the non-work-conserving discipline may be idle even when there are cells waiting for service, thus possibly increasing the average delay. However, the end-to-end delay bound is a more important performance index than average delay for guaranteed services [1]. In the existing algorithms, cells are immediately eligible for transmission upon their arrivals to a switch, and thus the existence of a perfect matching in which every input is matched to a unique output cannot be ensured in every time slot. As a result, although 100% throughput can be achieved asymptotically, no QoS guarantees can be provided. In SSF, the time axis is divided into constant periods of length T, called frames, each of which is an integer multiple of the cell transmission time. Cells arrived at the inputs of a switch during one frame are first held in the input buffers, and are then "sorted-and-transmitted" in the next frame. The (r, T) traffic model is adopted in SSF, in which a connection with a rate of r cannot transmit more than $r \cdot T$ bits during time T. Bandwidth allocation is performed before a connection is established to assign the connection rate in such a way that the aggregate rate of any link is below its capacity. The SSF algorithm consists of a cell admission policy to regulate the traffic pattern to conform to the (r, T) traffic model at the source node of each connection, and a sorting algorithm at each switching node to resolve the input and output contentions. It is proved that cells arrived during one frame can be transmitted in the next frame. Therefore, an input-queued switch employing SSF can achieve strict sense 100% throughput, and provide deterministic end-to-end delay and delay jitter bounds. Since SSF is a non-work-conserving scheduler, the performance analysis can be extended from a single node to a network with arbitrary topology, and more efficient usage of buffer space than work-conserving schedulers can be achieved [1].

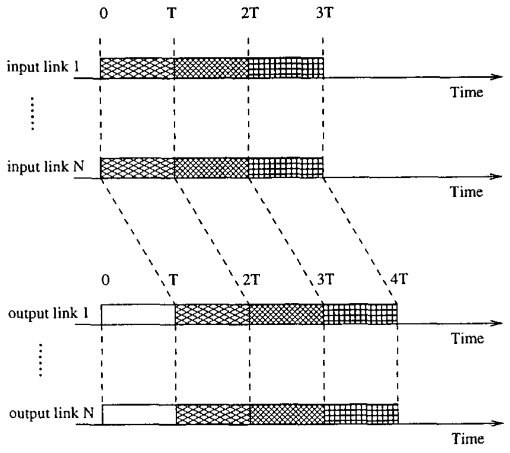

Fig. 2. Time relation between cell arrivals and departures

The rest of the paper is organized as follows. Section 2 presents the SSF algorithm. The proof of the QoS guarantees and the analysis of the complexity of SSF are given in Section 3. Concluding remarks are given in Section 4.

II. The Store-Sort-and-Forward algorithm

The Store-Sort-and-Forward (SSF) algorithm consists of two parts: a cell admission policy which is only needed at the source node of each connection to regulate the traffic to conform to the (r, T) traffic model [14], and a sorting algorithm, which is needed at each switching node to resolve input and output contentions.

A. Framing strategy and cell admission policy

Consider an $N \times N$ input-queued cell switch in which buffers are only placed at the input side of the switch. A framing strategy similar to [14] is adopted in SSF. At each switch, the time axis is divided into frames with equal length of T, where T is an integer multiple of the transmission time of a cell. We assume that all input and output links have the same transmission rate R and are synchronized, i.e., input and output links start service at the same point of time, as shown in Fig. 1. For simplicity, the frame size M is expressed in unit of cells, i.e., $M = \frac{1}{L} \cdot R \cdot T$, where L is the cell length in bits.

Cells arrived during one frame are first held in the input buffers and eligible for transmission in the next frame, as shown in Fig. 2. Cells stored in one input buffer may go to any of the outputs. Let $r_{i,j}$ be the rate of a connection between input i and output j. The traffic load of any input or output link should be kept below its capacity to avoid cell loss, i.e., $\sum_{i=1}^{N} r_{i,j} \cdot T \leq R \cdot T$ and $\sum_{j=1}^{N} r_{i,j} \cdot T \leq R \cdot T$. Thus,

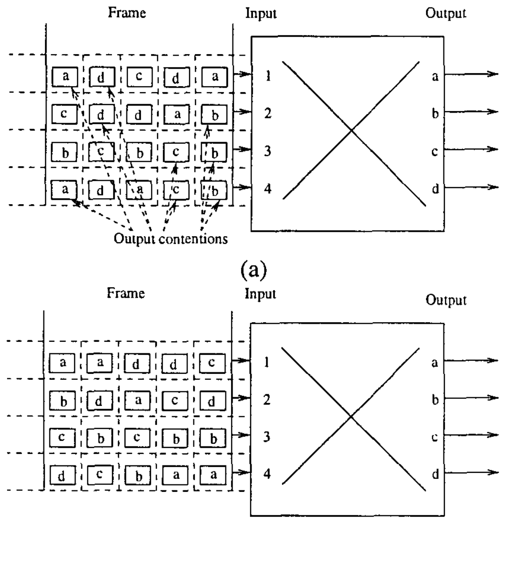

Fig. 3. Cells arrived in one frame: (a) original arrival orders. (b) scheduled transmission orders.

bandwidth allocation must be performed in the signaling phase before a connection is established. To guarantee that the traffic on any link is not overloaded, a cell admission policy is needed to regulate the traffic to conform to (r, T) traffic model, in which a connection with a rate of $r_{i,j}$ can transmit no more than $r_{i,j} \cdot T$ bits during a frame with length of T. If each server along a connection path guarantees that cells arrived during one frame can always be transmitted in the next frame, the connection will conform to the (r, T) traffic model at every switch throughout the network [14], and therefore, it is only necessary to have the cell admission function at the source node of the connection.

B. The sorting algorithm

The sorting algorithm is a key element of SSF to ensure that cells arrived in one frame can be transmitted in the next frame by completely resolving the input and output contentions. Cells arrived at one input during a frame can go to any of the outputs. Since there is no speedup at either the buffers or the fabric, only one cell can be transmitted from each input, and likewise only one cell can be forwarded to each output in any given time slot. Thus, contentions occur when more than one cell is destining for the same output in the same time slot, as shown in Fig. 3(a). Contention is the main problem which restricts an input-queued switch from providing QoS guarantees, and can be resolved by rearranging the transmission order of the cells among different connections[2] arrived during one frame in each input buffer in such a way that cells to be transmitted in the same time slot are going to different outputs, as shown in Fig. 3(b). Let $C_{i,j}$ be the number of cells which arrive at input i and are directed to output j in one frame. We will prove in Section 3 that such a rearrangement always exists if the traffic pattern conforms to the (r, T) model and the traffic is not overloaded, i.e., $\sum_i C_{i,j} \leq M$ and $\sum C_{i,j} \leq M$. For simplicity, we make the following assumption.

Assumption 1: [3] All the input and output links are fully utilized, i.e.,

$$\sum_i C_{i,j} = M \quad and \quad \sum_j C_{i,j} = M \quad \forall i, j = 1, 2, \cdots, N.$$

Definition 1: A perfect matching is a matching in which every input is matched to a unique output.

To ensure that cells arrived in one frame can be forwarded to the output links in the next frame, a perfect matching is needed in each time slot. A traffic matrix $W = [w_{i,j}]$ can be generated for each frame, where every row i in W represents a distinct input k, and every column j represents a distinct output l. Note that i is not necessarily equal to k; likewise, j may not equal to l. The element $w_{i,j}$ at the intersection of row i and column j indicates the number of cells destined for the corresponding output l from the corresponding input k. The sum of elements along any row or column is exactly M based on Assumption 1. If elements along the diagonal of the matrix are all nonzero, every input is matched to a unique output. Such a matrix, referred to as a matched matrix, is attainable by swapping rows and columns of an arbitrary traffic matrix if Assumption 1 is held. A modified version of McWorter's algorithm which was originally proposed to resolve the marriage problem [15] is used to obtain a matched matrix.

Consider a traffic matrix which can be decomposed into the following form:

$$W_0 = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where blocks A and D are square, and all diagonal elements of A are nonzero[4].

Definition 2: An effective move consists of row and column swappings with the constraint that all diagonal elements of A remain nonzero that will result in

- moving a nonzero element in D to the upper-left corner of D,
- replacing a zero element in B by a nonzero element in A, or
- replacing a zero element in D by a nonzero element in C.

A matched matrix can thus be obtained by the following iterative procedure in each time slot:

1. **Single effective move:** If there is at least one nonzero element in block D, that element can be moved to the upper-left corner of D by swapping rows and columns without

[2]Changing the relative transmission order of cells belonging to an individual connection does not affect the contentions.

[3]If $\sum_i C_{i,j} < M$ or $\sum_j C_{i,j} < M$, void cells can be easily inserted into corresponding input buffers at the end of each frame to meet Assumption 1. Void cells are sorted as normal cells, but are not transmitted.

[4]Initially, the size of A can be 0.

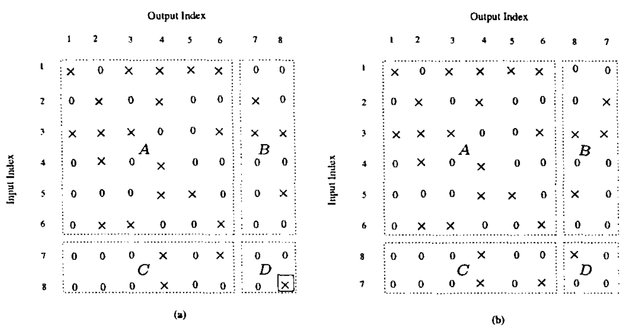

Fig. 4. **A single effective move:** (a) There is a nonzero element in block D. (b) The nonzero element is moved to the upper-left corner of block D by swapping rows 7 and 8, and then columns 7 and 8.

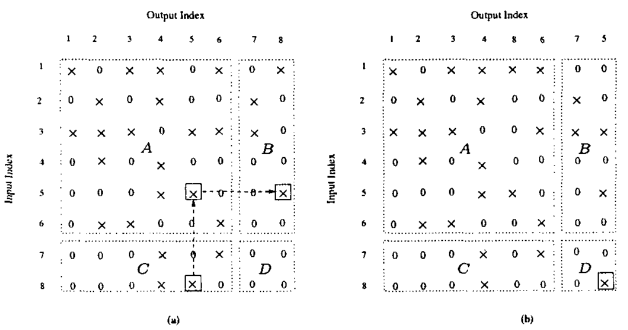

Fig. 5. **A double effective move:** (a) Block D has only zeros and there is a nonzero element in block B corresponding to a nonzero element in block C. (b) The nonzero element in block C is moved to block D by swapping columns 5 and 8.

changing any element in block A, as shown in Fig. 4. Therefore, the size of A is increased by one[5] within a single effective move. This step is repeated until all elements in D become zero or the size of D becomes zero.

2. **Double effective move:** If all elements in D become zero, find a nonzero element in C. There must be some nonzero elements in block C, otherwise the input buffers corresponding to the rows in blocks C and D do not contain any cell, thus violating Assumption 1. For any nonzero element $w_{i,j}$ in block C, if there exists a corresponding nonzero element $w_{j,k}$ for some k in block B, interchange columns j and k, thus resulting in one nonzero element in block D, as shown in Fig. 5. By repeating Step 1, the size of A is increased by one. That is, it takes two effective moves to increase the size of A by 1, thus so called a **double effective move**.
3. **Multiple effective move:** For every element $w_{i,j}$ in block C, if the corresponding element $w_{j,k}$ is zero for all k in block B, rows and columns are swapped (see Fig. 6(b)) such that nonzero columns of C are moved right next to block D resulting in the following matrix:

$$W_1 = \begin{bmatrix} A_1 & X & G \\ F & A_2 & 0 \\ 0 & E & D \end{bmatrix},$$

where block E contains all the nonzero columns of C and all diagonal elements of A_1 and A_2 are still nonzero. Block F is called a residue matrix. We will prove in Section 3 that elements in block F cannot be all zero. For any nonzero element $w_{i,j}$ in F, if there exists a corresponding nonzero element $w_{j,k}$ for some k in G, interchange columns j and k, thus resulting in a nonzero element in the zero block just below G (i.e., completion of one effective move), as shown in Fig. 6(c). Then, Step 2 can be repeated to augment the size of A by one. However, for every nonzero element $w_{i,j}$ in F, if the corresponding element $w_{j,k}$ is zero for all k in block G, the sub-matrix consisting of all but the extended rows and extended columns[6] of block E satisfies the same conditions as the original situation in Step 3, and can thus be further decomposed as above. Hence, either a nonzero element can be moved to the zero block below G or the traffic matrix can be recursively decomposed.

We will prove in Section 3 that a perfect matching can be found within $N(\ln N + O(1))$ effective moves if the traffic pattern conforms to Assumption 1. Once a matched matrix is found in a time slot, a cell corresponding to each diagonal elements, say $w_{i,i}$, is scheduled to be transmitted during the next frame from the input corresponding to row i to the output corresponding to column i. Then the value of each diagonal element is reduced by one. In the next time slot, a new matched matrix is obtained based on the updated traffic matrix until all the cells in the frame are scheduled.

III. Algorithm analysis

In this section, we first prove that by rearranging the cell transmission orders in each input buffer, a contention free schedule can be found in each time slot if the traffic pattern follows Assumption 1, and then we analyze the complexity of the algorithm.

A. Guaranteed QoS

Finding a perfect matching is the same as finding a system of distinct representatives (SDR) in a family of sets [16].

Definition 3: Suppose $\Omega = \{S_1, S_2, \cdots, S_N\}$ is a family of sets, where S_i, $i = 1, 2, \cdots, N$, is a set of elements. Sets S_i and S_j are not necessarily distinct $\forall i \neq j$. Let $\underline{V} = (a_1, a_2, \cdots, a_N)$ be an N-tuple vector with $a_1 \in S_1$, $a_2 \in S_2$, $\cdots$, $a_N \in S_N$. Then, $\underline{V}$ is called a system of representatives

[5]We adopt the notation that the size of an $N \times N$ matrix when increased by one becomes an $(N+1) \times (N+1)$ matrix.

[6]An extended row of a block is referred to as the row of the traffic matrix which contains the row of that block; an extended column of a block is referred to as the column of the traffic matrix which contains the column of that block.

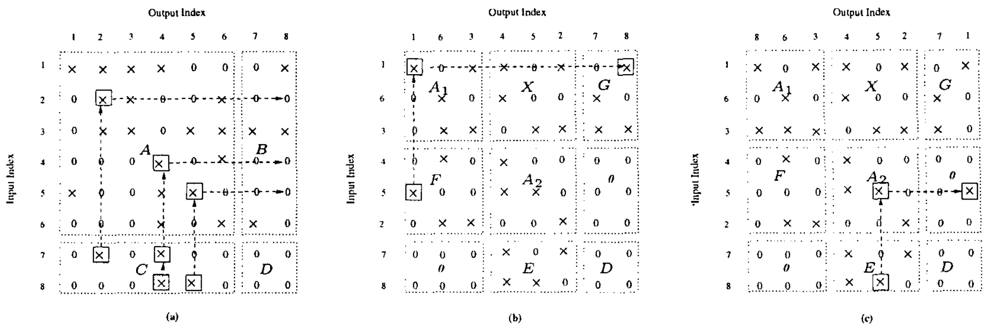
Fig. 6. **A multiple effective move:** (a) There are only zero elements in block B corresponding to the nonzero elements in block C. (b) Residue matrix F is constructed by swapping columns 2 and 6, and then rows 2 and 6. There is a nonzero element in G corresponding to a nonzero element in F. (c) The nonzero element in F can be moved to the zero block below G by swapping columns 1 and 8.

for Ω. In addition, if a_i's are all distinct, $\underline{V}$ is called a system of distinct representatives (SDR) for Ω.

Fact 1: Philip Hall's Theorem [16]: The family of sets $\Omega = \{S_1, S_2, \cdots, S_N\}$ possesses an SDR iff for all $k = 1, 2, \cdots, N$, a union of any k sets in Ω, $\overset{k}{\cup} S_i$, contains at least k distinct elements, i.e., $|\overset{k}{\cup} S_i| \geq k$, where $|S|$ represents the number of distinct elements in S.

Let I_i be a set, where elements in I_i represent the distinct destinations of the cells arrived at input i in one frame. Thus, a perfect matching is in fact an SDR of the family of sets $I = \{I_1, I_2, \cdots, I_N\}$.

Lemma 1: Let $\overset{k}{\cup} I_i$ be a union of any k sets in I. The following relation holds for any traffic pattern which satisfies Assumption 1:

$$|\overset{k}{\cup} I_i| \geq k,$$

i.e., the number of distinct destinations of cells belonging to one frame in k input buffers is at least k.

Proof: This is proved by contradiction. Assume that the statement is not true, i.e., the number of distinct destinations of cells belonging to one frame in any k input buffers could be less than k. Since the total number of cells belonging to the frame in these k input buffers is $k \cdot M$, at least one output link must be receiving more than M cells, thus violating Assumption 1. Therefore, the Lemma must be true. □

Let $O = \{O_1, O_2, \cdots, O_N\}$ be a family of sets, where O_i is a set, whose elements are the distinct inputs from which cells belonging to one frame are directed to output i.

Lemma 2: Let $\overset{k}{\cup} O_i$ be a union of any k sets in O. The following relation holds for any traffic pattern which satisfies Assumption 1:

$$|\overset{k}{\cup} O_i| \geq k,$$

i.e., the number of distinct sources (inputs) of the cells received by k outputs in one frame is at least k.
The proof is the similar to that of Lemma 1. □

Lemma 3: Let Γ be a set of any k inputs, and Θ be a set of any k outputs, where $k = 1, 2, \cdots, N$. If cells belonging to one frame in the k input buffers in Γ are directed to the k outputs in Θ exclusively, the k outputs in Θ can only receive cells from these k inputs in Γ during the frame given that the traffic pattern satisfies Assumption 1, and vice versa.

Proof: This is proved by contradiction. Assume that the statement is not true, i.e., under the condition that cells belonging to one frame in k input buffers in Γ are directed to the k outputs in Θ exclusively, the k outputs in Θ can still receive cells from inputs which are not in Γ. According to Assumption 1, the number of cells which can be transmitted to any k outputs or from any k inputs in each frame is exactly $k \cdot M$. If the k outputs in Θ receive cells from inputs which are not in Γ, the number of cells transmitted to these k outputs from the k inputs in Γ must be less than $k \cdot M$. Thus, some of the cells in the k inputs in Γ must be transmitted to some outputs not in Θ, which contradicts our condition, and therefore, the lemma must be true. Similarly, if cells received by the k outputs in Θ all come from the k inputs in Γ during one frame, the cells stored in these k input buffers in Γ can only be directed to the k outputs in Θ in that frame. □

Lemma 4: A perfect matching always exists if the traffic pattern satisfies Assumption 1.

Proof: This lemma follows directly from Lemma 1. Since $|\overset{k}{\cup} I_i| \geq k$ if the traffic pattern satisfies Assumption 1, according to Philip Hall's Theorem, an SDR exists for the family of sets I, which is in fact a perfect matching. □

Theorem 1: SSF can guarantee strict sense 100% throughput, bounded end-to-end delay, and bounded end-to-end delay jitter.

Proof: From Lemma 4, a perfect matching exists provided Assumption 1 is satisfied, i.e.,

$$\sum_i C_{i,j} = M \quad and \quad \sum_j C_{i,j} = M \quad \forall i, j = 1, 2, \cdots, N,$$

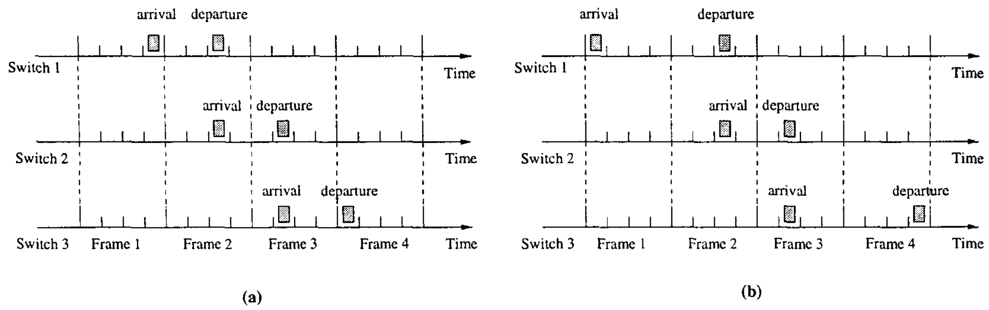

Fig. 7. Two extreme cases of cell transmission: (a) the cell experiences an end-to-end delay of $(n-1)T+\tau$. (b) the cell experiences an end-to-end delay of $(n+1)T-\tau$.

where $C_{i,j}$ is the number of cells which arrive at input i and directed to output j in one frame.

Once a perfect matching is found in a time slot, the rest of the cells belonging to that frame still satisfy Assumption 1, except that instead of M there are $M-1$ cells left in each input buffer. Therefore, a perfect matching can be found in each time slot until all the cells belonging to that frame are scheduled, and thus 100% throughput is achieved.

Noting that cells arrived during one frame can be transmitted in the next frame, the following are two extreme cases of cell transmission for a connection consisting of n concatenated switches: a cell arrived in the last time slot of frame f at the first switch is transmitted in the first time slot of frame $f+n$ at the nth switch resulting in an end-to-end delay of $(n-1)T+\tau$, where τ is the time duration of one time slot, and a cell arrived in the first time slot of frame f at the first switch is transmitted in the last time slot of frame $f+n$ at the nth switch resulting in an end-to-end delay of $(n+1)T-\tau$, as shown in Fig. 7. Thus, the end-to-end delay and end-to-end delay jitter of a connection are bounded by $(n+1)T-\tau$ and $2(T-\tau)$, respectively. □

B. Complexity issues

In this section, we derive the upper bound of the number of effective moves needed to obtain a perfect matching.

Consider the following $N \times N$ traffic matrix satisfying Assumption 1,

$$W_0 = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where block A is a $P \times P$ block, D is an $(N-P) \times (N-P)$ block, and all diagonal elements of A are nonzero.

The rows and columns in W_0 are associated with the inputs and outputs, respectively. Assume that row i is corresponding to input k, and column j is corresponding to output l, where $1 \leq i, j, k, l \leq N$. Therefore, a nonzero element $w_{i,j}$ indicates that input k has cells directed to output l.

Lemma 5: If D is an $(N-P) \times (N-P)$ "zero" block, the number of nonzero columns in block C is at least $N-P+1$, and similarly, the number of nonzero rows in block B is at least $N-P+1$.

Proof: There are $N-P$ rows in blocks C and D. According to Lemma 1, cells stored in the inputs corresponding to these rows are directed to at least $N-P$ distinct outputs, i.e., there are at least $N-P$ nonzero columns in blocks C and D. Since D is a "zero" block, there are at least $N-P$ nonzero columns in block C. However, if the number of nonzero columns in block C is $N-P$, i.e., cells stored in the inputs corresponding to the $N-P$ rows are directed to $N-P$ distinct outputs, according to Lemma 3, the $N-P$ outputs corresponding to the $N-P$ nonzero columns in block C can only receive cells from these $N-P$ inputs. Hence, columns in block A corresponding to these $N-P$ outputs must be zero, thus conflicting with the condition that the diagonal elements of A are all nonzero. Therefore, the number of nonzero columns in block C is at least $N-P+1$. Similarly, the number of nonzero columns in block B is at least $N-P+1$. □

In the previous section, we claimed that block F in the traffic matrix W_1 must have some nonzero elements. The proof is given by the following Lemma. Note that in the scenario of a multiple effective move, the traffic matrix is decomposed recursively, and a residue matrix is constructed in each iteration of the recursive procedure until a nonzero element $w_{j,k}$ corresponding to a nonzero element $w_{i,j}$ in the residue matrix is found in block G, as shown in Fig. 8.

Lemma 6: Consider a traffic matrix which satisfies Assumption 1. Any residue matrix F associated with an $(N-P) \times (N-P)$ all-zero block D must have at least $N-P+1$ nonzero columns.

Proof: Consider the residue matrix constructed in the first iteration of the recursive procedure. For convenience, the traffic matrix W_1 is written again below:

$$W_1 = \begin{bmatrix} A_1 & X & G \\ F & A_2 & 0 \\ 0 & E & D \end{bmatrix},$$

where D is an $(N-P) \times (N-P)$ "zero" block, and diagonal elements of A_1 and A_2 are all nonzero. By virtue of the procedure in obtaining a matched matrix, the block below a residue matrix is always a "zero" block, and the extended rows of block F constitute a zero block in the area above D.

First, we prove that there are at least $N - P + 1$ **columns** in block F. According to Lemma 5, there are at least $N - P + 1$ nonzero columns in block C and $N - P + 1$ nonzero rows in block B. Thus, if the number of zero columns in block C is less than $N - P + 1$, there must exist at least one nonzero element $w_{j,k}$ in block B corresponding to a nonzero element $w_{i,j}$ in block C, which results in a double effective move, and no residue matrix is needed to be constructed. Thus, residue matrix F is only necessary when the number of zero columns in C is larger than or equal to $N - P + 1$, i.e., there are at least $N - P + 1$ **columns** in block F.

Let m be the number of columns in F, i.e., block A_1 is an $m \times m$ block, and n be the number of nonzero columns in F, and thus there are $m - n$ zero columns in F. Suppose cells received by the outputs corresponding to these $m - n$ extended columns of F and the $N - P$ extended columns of G come from d distinct inputs. Since all nonzero elements in these extended columns are in blocks A_1 and G, the number of distinct inputs d is no more than m, i.e., $d \leq m$.

Consider $n < N - P$: By Lemma 2, since $m - n + N - P > m$ for $n < N - P$, the number of distinct inputs d corresponding to these $m - n + N - P$ outputs must be greater than or equal to $m - n + N - P$, i.e., greater than m. This contradicts the above, and thus $n \geq N - P$.

Consider $n = N - P$: By Lemma 2, since $m - n + N - P = m$ for $n = N - P$, the number of distinct inputs d corresponding to these $m - n + N - P$ outputs must be greater than or equal to $m - n + N - P$, i.e., $d \geq m$. On the other hand, d must be less than or equal to m as shown above, which implies that d must be equal to m. However, by Lemma 3, the cells stored in these $d = m$ inputs corresponding to the rows in A_1 are exclusively directed to the m outputs corresponding to the $m - n$ columns in block A_1 and the $N - P$ columns in block G, thus resulting in n zero columns in block A_1. This conflicts with the condition that all diagonal elements in A_1 are nonzero. Therefore, n is at least $N - P + 1$.

Noting that the elements below any residue matrix and the elements below block G are always zero, the proof of the lemma for a residue matrix constructed in any other iteration of the recursive procedure is similar to that for the residue matrix constructed in the first iteration, and thus the lemma is proved. □

Theorem 2: Let k, where $k \geq 1$, be the number of effective moves within which the size of block A is guaranteed to increase from $P \times P$ to $(P + 1) \times (P + 1)$ in SSF, where $P = 0, 1, 2, \cdots, N - 1$. The following relation between k and P holds for any traffic patterns satisfying Assumption 1.

$$\frac{(k-1)\cdot(N+1)}{k} \leq P < \frac{k\cdot(N+1)}{k+1}.$$

Proof: According to Lemma 6, the number of nonzero columns in a residue matrix is at least $N - P + 1$ if the traffic pattern satisfies Assumption 1. The larger the number, the smaller number of decompositions of the traffic matrix is needed to guarantee that a nonzero element $w_{j,h}$ in G corresponding to a nonzero element $w_{i,j}$ in the residue matrix can be found, i.e., a smaller number of effective moves are needed. To derive the number of effective moves within which the size of block A is guaranteed to increase by one, the worst case is considered. Thus, we assume that there are $N - P + 1$ nonzero columns in each residue matrix as shown in Fig. 8.

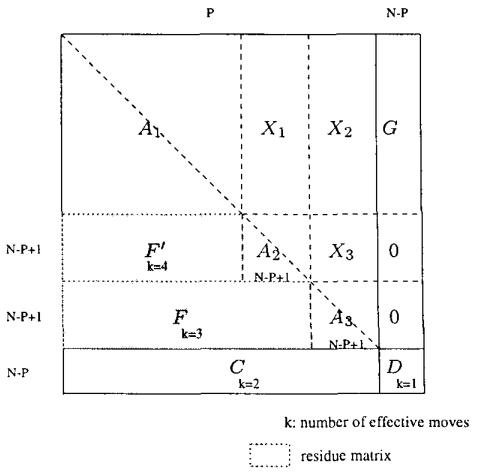

Fig. 8. Decomposition of a traffic matrix

Note that the size of a residue matrix is $(N - P + 1) \times (P - (k-2)(N - P + 1))$, where k is the number of effective moves required to move a nonzero element to the upper-left corner of block D if a nonzero element $w_{j,h}$ which is corresponding to a nonzero element $w_{i,j}$ in the residue matrix can be found in G, as shown in Fig. 8. Since there are $N - P + 1$ nonzero rows in G and $N - P + 1$ nonzero columns in a residue matrix, if the number of columns in the residue matrix is less than $2(N - P + 1)$, i.e., $P - (k - 2)(N - P + 1) < 2(N - P + 1)$, or equivalently, $P < \frac{k\cdot(N+1)}{k+1}$, at least one nonzero element $w_{j,h}$ corresponding to a nonzero element $w_{i,j}$ in that residue matrix can be found in G. Thus, k effective moves are sufficient to increase the size of block A by one if $P < \frac{k\cdot(N+1)}{k+1}$. Similarly, $k-1$ effective moves are sufficient to increase the size of block A by one if $P < \frac{(k-1)\cdot(N+1)}{k}$. Thus, k effective moves are necessary only if $\frac{(k-1)\cdot(N+1)}{k} \leq P$. Therefore, k effective moves are necessary and sufficient to guarantee to increase the size of block A by one if $\frac{(k-1)\cdot(N+1)}{k} \leq P < \frac{k\cdot(N+1)}{k+1}$. □

Theorem 3: The number of effective moves required to obtain a perfect matching in SSF is bounded by $N(\ln N + O(1))$.

Proof: From Theorem 2, the size of block A is guaranteed to increase by one within k effective moves if the number of rows (columns) in block A satisfies the following condition,

$$\frac{(k-1)\cdot(N+1)}{k} \leq P < \frac{k\cdot(N+1)}{k+1} \qquad (1)$$

Equation (1) can be rewritten as follows:

$$\frac{P}{N-P+1} \leq k \leq \frac{N+1}{N-P+1}$$

Thus, the total number of effective moves $f(N)$ required to obtain a perfect matching in the worst case is $\sum_{P=0}^{N-1} k$ which is bounded by

$$f(N) \quad = \quad \sum_{P=0}^{N-1} k \leq \sum_{P=0}^{N-1} \frac{N+1}{N-P+1} \qquad (2)$$

Rewrite the right part of Equation (2) in terms of n, where $n = N - P + 1$.

$$f(N) \quad \leq \quad (N+1) \sum_{n=2}^{N+1} \frac{1}{n} \leq N \sum_{n=1}^{N} \frac{1}{n} \qquad (3)$$

Note that the sum of the Harmonic series is given by:

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$$

Thus,

$$f(N) \quad \leq \quad N(\ln N + O(1)) \qquad (4)$$

i.e., the number of effective moves required to obtain a perfect matching is bounded by $N(\ln N + O(1))$. □

IV. Conclusions

In this paper, a novel scheduling algorithm, referred to as Store-Sort-and-Forward (SSF), has been proposed to provide QoS guarantees for input-queued switches. A framing strategy is adopted in the SSF algorithm, in which the time axis is divided into fixed length frame. SSF adopts a non-work-conserving discipline, i.e., cells arrived during one frame are only eligible for transmission in the next frame. By incorporating a bandwidth allocation strategy and a cell admission policy, SSF ensures that cells arrived at a switch during one frame can be transmitted in the next frame. Therefore, strict sense 100% throughput is guaranteed, and the end-to-end delay time and delay jitter are bounded. It has also been proved that a perfect matching can be obtained within $N(\ln N + O(1))$ effective moves.

References

[1] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.

[2] Ascend Communications, *GRF Family of Switches*, www.ascend.com.

[3] Digital Equipment Corporation, *GIGA switch*, www.networks.digital.com.

[4] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE INFOCOM*, 1996, pp. 296–302.

[5] N. McKeown, J. Walrand, and P. Varaiya, "Scheduling cells in an input-queued switch," *IEE Electronics Letters*, pp. 2174–2175, Dec. 9th 1993.

[6] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in *Proceedings of the ICCCN*, Oct. 1996, pp. 226–231.

[7] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1347–1356, Dec. 1987.

[8] M. J. Karol, K. Y. Eng, and H. Obara, "Improving the performance of input-queued ATM packet switches," in *Proceedings of IEEE INFOCOM*, 1992, pp. 110–115.

[9] H. Obara and T. Yasushi, "An efficient contention resolution algorithm for input queueing ATM cross-connect switches," *Internation Jour. of Digital & Analog Cabled systems*, vol. 2, no. 4, pp. 261–267, Oct.-Dec. 1989.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw Hill, New York, 1989.

[11] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," Tech. Rep. CSL-TR-98-758, Stanford University, 1998.

[12] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," in *Proceedings of IWQoS*, 1998, pp. 218–224.

[13] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *IEEE Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, Nov. 1993.

[14] S. J. Golestani, "A stop-and-go queueing framework for congestion management," in *Proceedings of the ACM SIGCOMM*, 1990, pp. 8–18.

[15] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, Massachusetts, 1986.

[16] F. S. Roberts, *Applied Combinatorics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.