On the Speedup Required for Work-Conserving Crossbar Switches

Pattabhiraman Krishna, Naimish S. Patel, Anna Charny, and Robert J. Simcoe

Abstract— **This paper describes the architecture for a work-conserving server using a combined I/O-buffered crossbar switch. The switch employs a novel algorithm based on output occupancy, the lowest occupancy output first algorithm (LOOFA), and a speedup of only two. A work-conserving switch provides the same throughput performance as an output-buffered switch. The work-conserving property of the switch is independent of the switch size and input traffic pattern. We also present a suite of algorithms that can be used in combination with LOOFA. These algorithms determine the fairness and delay properties of the switch. We also describe a mechanism to provide delay bounds for real-time traffic using LOOFA. These delay bounds are achievable without requiring output-buffered switch emulation.**

Index Terms— **Crossbar switches, quality-of-service (QoS), scheduling, work-conservation.**

I. Introduction

The rapid growth in the popularity of the Internet has caused the traffic on the Internet to double every year for the last several years. It has also spurred the emergence of many Internet service providers (ISP's) whose revenues primarily come from providing Internet access to individuals and corporate customers. The ISP's typically lease wide-area links that cost them, for some very high capacity links (e.g., OC-48), up to several dollars per minute. In order to remain profitable in such an environment, the ISP's need to keep these links fully utilized. There is a dire need to reduce/eliminate link idling. Also, to sustain growth, they need to provide new differentiated services—e.g., tiered service, support for multimedia applications, etc.

The switches/routers in the ISP's networks play a critical role in providing these features. It is desirable that mechanisms be built into these switches so that they can be work conserving (to eliminate link idling), support prioritization (for tiered service), and provide rate and delay guarantees (to support multimedia applications). In addition, the switch has to be of a very high capacity.

Optimal throughput and delay performance is obtained using output-buffered switches. As long as input ports and output ports are undersubscribed (i.e., feasible loads), 100% throughput is achieved. Moreover, since the packets are immediately placed in the output buffers upon arrival, it is possible to better control the latency of the packet. This helps in providing QoS guarantees. To achieve this, the switch fabric must operate at a rate at least equal to the aggregate of all the input links connected to the switch. However, increasing line rates (B) and increasing switch size (N) make it extremely difficult to significantly speed up the switch fabric and also build memories with a bandwidth of $O(NB)$. This has renewed interest in switches with lower complexity (and cost), such as input-buffered switches.

One of the most popular interconnection networks used for building input-buffered switches is the crossbar because of its 1) low cost; 2) good scalability; and 3) nonblocking properties. An input-buffered crossbar switch has the crossbar fabric running at the link rate. If each input maintains a single first in/first out (FIFO) queue, packets suffer from head-of-line (HOL) blocking. This limits the maximum throughput achievable. Karol *et al.* [1] showed that the maximum throughput of an input-buffered crossbar switch operating under uniform traffic is limited to about 58%. Moreover, Li [2] has shown that the maximum throughput of the switch decreases monotonically with increasing burst size. A considerable amount of work has been done in recent years to build input-buffered switches that match the performance of an output-buffered switch. To eliminate HOL blocking, virtual output queues (VOQ's) [5], [7] were proposed at the inputs. Each input has N queues, one for each output, resulting in a total of N^2 queues. However, since there could be contention at the inputs and outputs, there is a necessity for an arbitration algorithm to schedule packets between various inputs and outputs (equivalent to the matching problem for bipartite graphs).

Definition 1: A maximal match on a bipartite graph is one where no more matches can be made trivially. A maximum match on the other hand is one that matches the maximum number of inputs and outputs, i.e., there is no other match that matches more inputs and outputs. A maximum match is maximal; however, the reverse is not true.

It has been shown that an input-buffered switch with VOQ's can provide asymptotic 100% throughput using a maximum matching algorithm [3]. However, the complexity of the best known maximum match algorithm is too high $[O(N^{2.5})$ for

Manuscript received July 15, 1998; revised February 1, 1999. An early version of this paper was presented at IWQoS '98, Napa Valley, CA, May 1998.

P. Krishna is with the Compaq Computer Corp., Marlboro, MA 01752 USA (e-mail: pat_krishna@yahoo.com).

N. S. Patel was with the Digital Computer Corp., Littleton, MA 01460 USA. He is now with Sycamore Networks, Tewksbury, MA 01876 USA (e-mail: naimish@sycamorenet.com).

A. Charney was with the Digitial Computer Corp., Littleton, MA 01460 USA. She is now with Cabletron Systems, Andover, MA 01810 USA (e-mail: charny@ctron.com).

R. J. Simcoe was with the Digitial Computer Corp., Littleton, MA 01460 USA. He is now with Nexabit Networks, Marlboro, MA 01752 USA (e-mail: bsimcoe@nexabit.com).

Publisher Item Identifier S 0733-8716(99)04486-8.

uniform traffic and $O(N^3 \log N)$ for nonuniform traffic] [4] for high-speed implementations. Moreover, under certain traffic conditions, maximum matching can lead to starvation. Over the years, a number of maximal matching algorithms (e.g., PIM [5], WPIM [6], SLIP [7], FARR [8], and LPF [12]) have been proposed. However, none of these algorithms match the performance of an output-buffered switch.

Increasing the *speedup* of the switch fabric has also been proposed as one of the ways to improve the performance of an input-buffered switch. Speedup is defined as the ratio of the switch-fabric bandwidth and the bandwidth of the input links. (Until otherwise mentioned, we will be assuming that all input links and output links of the switch have the *same* capacity.) However, when the switch fabric has a higher bandwidth than the links, buffering is required at the outputs too. Thus, a combination of input-buffered and output-buffered switches is required [called combined input- and output-buffered (CIOB) switches]. The goal then, is to find the minimum speedup required to match the performance of an output-buffered switch.

One of the early efforts in this direction was the Knockout Switch [16] proposed by Yeh *et al.* Instead of speedup, they employed a switch with a *parallelism factor* L, where L is the maximum number of packets that can be transferred to each output in a time slot. In [17] and [18], it was observed that more than 99.5% throughput is achievable using $L = 4$.

Unlike a switch operating under a *speedup* of L, in a switch with a *parallelism factor* of L, the L packets to an output cannot come from the same input. In [11], a simulation study of input-buffered switches suggested that a speedup of two is sufficient to provide both throughput and delay performance of an output-buffered switch. However, there were no proofs to back their claims. McKeown *et al.* [9] showed that a CIOB switch with VOQ's is always work conserving if speedup is greater than $N/2$. Prabhakar *et al.* [10] showed that a speedup of four is sufficient to emulate an output-buffered switch (with an output FIFO) using a CIOB switch with VOQ's. In a recent work [19], it has been shown that a speedup of two is sufficient to emulate an output-buffered switch employing a monotonic and work-conserving scheduler. As explained later in Section VI, the proposed form of the algorithm in [19] is too complex to implement using the current technology. The implementation difficulty, however, does not dilute the theoretical importance of the result in any way.

In this paper, we present a novel and implementationally simple scheduling algorithm, called the lowest occupancy output first algorithm (LOOFA). We prove that a CIOB switch with VOQ's operating under LOOFA and a speedup of two is work conserving at all times. The work-conserving property of a switch operating under LOOFA is independent of the switch size and input traffic pattern. We also present a suite of algorithms that can be used in combination with LOOFA. These algorithms determine the delay and fairness properties of the switch. In this paper, we will also present a mechanism that, when augmented with a switch operating under LOOFA and speedup of two, cannot only provide work-conserving properties, but also provide delay guarantees. The arbitration delay is the key component in the end-to-end switch delay in a crossbar-based switch, partly because it is the one that is the most difficult to control. We derive expressions for the arbitration and output queueing delay bounds for a LOOFA switch. To our knowledge, LOOFA is the only arbitration algorithm that is not only simple to implement, but also provides both work-conserving properties and delay bounds in a crossbar switch for a speedup of only two. The delay bound obtained is in the same order as an emulated output-buffered switch (with an output FIFO) [10].

The rest of the paper is organized as follows. Section II presents the high-level description of the switch architecture used by LOOFA. Section III explains the problem solved by the proposed algorithm. Section IV describes the algorithm. Section V describes the queueing architecture required to provide QoS guarantees. Also presented are the delay bounds obtained using LOOFA. Section VI compares the complexity of LOOFA with the algorithm presented in [19]. Conclusions are presented in Section VII, and the proofs are presented in the Appendix.

II. High Level Description of the Architecture[1]

The interconnection architecture is an $N \times N$ crossbar with a speedup of S, where N is the number of crossbar ports. The crossbar connections between the input and output ports are called channels. The variable-length packets are broken into fixed-sized cells before being transmitted across the crossbar. The cells are reassembled at the output of the switch. In practice, the cell size is chosen such that the arbitration and scheduling functions can be performed within the time it requires the cell to traverse the speeded-up crossbar.

Cell slot is the time required to transmit a cell across the input and output link and is equal to C/B, where C and B are the cell size and the link bandwidth, respectively. To simplify the discussions, we are going to assume that all packets are the same size as a cell. We use cell and packet interchangeably throughout the paper.

Each input-line card has N VOQ's, each corresponding to a crossbar output port. Packets upon arrival from the physical input link are stored in the memory, and a pointer to the packet is appended to the VOQ corresponding to the output channel of the packet. The switch has a central arbiter that executes the iterative maximal-matching algorithm described in Section IV. In each cell slot, the arbiter schedules and transfers at most S cells from an input and S cells to an output, where S is the speedup. The arbiter operates on the contents of all the VOQ's across the switch. During each phase, upon completion of the matching algorithm, the arbiter sends to each input channel I the identifier of the output channel to which the input channel I can send a cell. Once a cell is transmitted across the crossbar, the cell is stored in the memory at the output side of the switch, waiting to be transmitted onto the outgoing link.

[1]The queueing architecture described in this section is sufficient to explain the LOOFA algorithm. However, it should be noted that the architecture needs to be augmented in order to provide features like per-flow fairness and delay guarantees. In this paper, we will only describe the queueing architecture to provide delay guarantees (Section V-A).

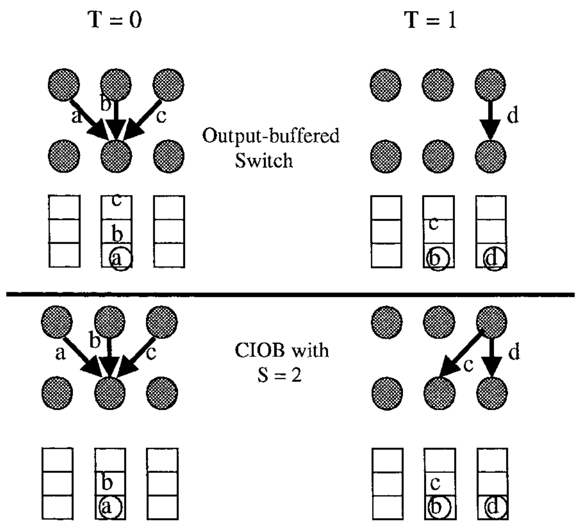


Fig. 1. An example of output-buffered emulation.

III. Problem

Each cell slot is divided into phases. For a speedup of S, there are S phases per cell slot. In each phase, an input can transfer at most one cell, and an output can receive at most one cell. Extreme values of $S = 1$ and $S = N$ represent an input-buffered switch and an output-buffered switch, respectively. For the intermediate values $1 < S < N$, we require buffering both at the input and the output, hence, the CIOB architecture. The problem that we try to solve is determining the *maximal*-matching algorithm that requires the smallest speedup to match the performance of an output-buffered switch, irrespective of the input-traffic pattern.

Fig. 1 illustrates an example of output-buffered emulation using a CIOB switch with $S = 2$. At $T = 0$, cells a, b, and c arrive at different inputs destined for output 2. At $T = 1$, cell d arrives at input 3 destined for output 3. The cells circled at the output are the cells that are currently being transmitted on the output link.

Identical behavior as an output-buffered switch means that under identical input traffic: (a) the CIOB switch is busy at the same time as the emulated switch; and (b) the packet departure order is the same at every output port. If only (a) is satisfied, then the throughput performance is matched, and if both (a) and (b) are satisfied, then delay performance is also matched. A work-conserving switch will satisfy condition (a). As can be observed in Fig. 1, both conditions (a) and (b) are satisfied by the example CIOB switch.

In this paper, we do not strive to exactly emulate an output-buffered switch. Instead we employ a very simple algorithm to provide work conservation and later augment the switch with a queueing architecture at the input to provide QoS guarantees. This enables us to build switches that are not only very high capacity, but also are work conserving and provide QoS guarantees.

IV. Description of the Algorithm

As stated earlier, for a speedup of S, there are S phases per cell slot. During each phase, an execution of the matching algorithm takes place. It is assumed that sufficient number of iterations is completed during the execution such that no more trivial matches can be added. For a switch of size N, N iterations are sufficient. New cell arrivals take place at the beginning of a cell slot. An input can receive at most one new cell per cell slot from its incoming link. Departures from the output take place at the end of a cell slot. An output can transmit at most one cell per cell slot to its outgoing link. As mentioned earlier, the input channels maintain a VOQ per output channel. Output channels maintain a queue to receive cells from input channels. An integer variable, named occupancy, is associated with each output queue. As the name implies, the occupancy of an output j at any time is simply the number of cells currently residing in output j's queue. The natural place to maintain occupancy values is at the arbiter. The arbiter keeps track of the number of times ($ncells$) an output was matched in the current cell slot. At the end of the cell slot, the arbiter increments the occupancy of an output by $ncells - 1$, simulating a cell departure from the output. However, if $ncells$ is zero and the occupancy is zero, the occupancy is not updated. Zero occupancy is permissible, implying that a single cell could have entered a zero-occupancy output and exited it during the cell slot.

Definition 2: A switch is work conserving if and only if each output of the switch for which there are cells (either at input or at the output) at the beginning of cell slot T is active at the end of the cell slot T.

Under a speedup of two, each cell slot has two phases. During each phase, the following steps occur.

1) Initially, all inputs and outputs are unmatched.
2) Each unmatched input selects the active VOQ (i.e., a VOQ that has at least one cell queued) going to an unmatched output with the lowest occupancy and sends a request to that output (*OUTPUT SELECTION*).
3) The output, upon receiving requests from multiple inputs, selects one and sends a grant to that input (*INPUT SELECTION*).
4) The switch returns to step 2 until no more connections can be made.

The algorithm essentially gives priority to output channels with low occupancy, thereby attempting to simultaneously maintain work conservation across all output channels. It is interesting that a crossbar speedup of only two is necessary and sufficient to ensure work conservation. The proof is presented in the Appendix.

The work-conserving feature of the switch is *independent* of the selection algorithm used at the outputs. Table I lists some examples of the selection algorithm that can be employed at the outputs.

The version of the algorithm mentioned earlier is the *greedy* one. In the *best-first* version, in each iteration, the inputs pick the lowest-occupancy unmatched output *across* the switch (i.e., global decision making, as opposed to local decision making in the greedy version) and performs step 3 for that output. If a unmatched input does not have any cell for the lowest-occupancy output, the input sends a null request to

TABLE I
EXAMPLE INPUT SELECTION ALGORITHMS

Input Selection Algorithm	Selection Criterion
Round-Robin	The output port selects the first input port that is pointed to by the round-robin pointer, or the next requesting input port beyond the round-robin pointer.
Oldest Cell First (OCF), Last Service Time	The output port selects the requesting input port with the oldest timestamp sent along with the request.
Random	The output port randomly selects an input port from one of the requesting input ports.

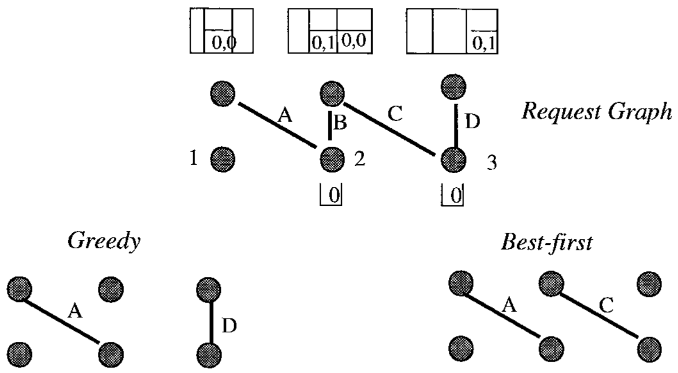


Fig. 2. Example illustrating the difference between greedy and best-first versions.

the arbiter. Thus, the best-first version of the algorithm has to execute N iterations to perform correctly. However, in the greedy version of the algorithm, multiple outputs can be matched in an iteration, thus requiring on average less than N iterations.

The example in Fig. 2 illustrates the difference between the matches made in the greedy version and best-first version of the algorithm. Each cell is represented by a tuple (OCC, TS), where OCC is the occupancy of the output of the cell, and TS is the timestamp assigned to the cell. For example, TS could be the time of arrival of the cell into the VOQ. The input-selection algorithm employed in this example is the *oldest cell first* (OCF). There are four cells in the request graph, A–D. For example, cell A has a TS of zero and is going to output 2, whose occupancy is zero. Thus, A is represented by (0,0). The inactive VOQ's are [1,1], [1,3], [2,1], [3,1] and [3,2], where $[x, y]$ represents a VOQ for output y in input x.

In the greedy version of the algorithm, in each iteration, each input sends a request to the lowest-occupancy unmatched output amongst the active VOQ's. Ties are broken by picking the lowest port number. Thus, inputs 1 and 2 will send a request to output 2, and input 3 will send a request to output 3. Using the OCF, output 2 selects input 1, and output 3 selects input 3. Thus, cells A and D are matched using the greedy version.

In the best version of the algorithm, in each iteration, each input sends a request to the lowest-occupancy unmatched output across the whole switch. In this example, the lowest-occupancy output across the switch is 2. Thus, inputs 1 and 2 will send a request to output 2, and since input 3 does not have any cell for output 2, it will send a null request in this iteration. In the first iteration, cell A gets matched. In the second iteration, the lowest occupancy unmatched output is 3. Inputs 2 and 3 send a request to output 3, which selects input 2 using OCF. Thus, cells A and C are matched using the best-first version.

It should be noted that *both* the greedy and the best-first version of the LOOFA algorithm are work conserving (see Appendix for the proofs). The choice of the input selection algorithm will determine the fairness properties in an overloaded switch, and delay properties under feasible rates. In this paper, we will only present the delay properties of the switch.

V. Providing Rate and Delay Guarantees Using LOOFA

In order to provide rate or delay guarantees for the QoS traffic, admission control is essential. In the absence of any admission control, the guarantees achievable for QoS traffic effectively converge to those of best-effort traffic. The QoS traffic also becomes less useful if the admitted flows are not policed somewhere, be it at the network edge or within the switch itself. In the absence of policing, flows that do not conform to their negotiated rates can cause problems to switches attempting to provide rate and delay guarantees to all admitted flows. In this paper, we will assume that the switch has to perform the policing itself. It is also implicitly assumed that the switch-control processor performs the admission control and the relevant per-flow parameters are communicated to the scheduling and arbitration modules in the switch.

In the absence of policing, and under the assumption that flows do in fact conform to their requested rate in long term average, the task of providing long-term rate guarantees is relatively simple. By admitting flows whose aggregate rate is less than the switch capacity (in particular, the aggregate rate of all flows to each output port must not exceed the port's capacity), one may simply use LOOFA and a speedup of two (which provides 100% throughput for feasible loads). No per-flow state needs to be maintained in the switch.

On the other hand, if the flows cannot be trusted to conform to their negotiated rates, some policing mechanism is needed to ensure that guarantees for each flow are met. An effective way to police traffic is to incorporate a rate controller that limits the maximum rate of a flow to its requested rate. An example implementation of a rate controller is the rate-controlled version of worst-case fair, weighted fair queueing (WF^{2}Q) [13]. The use of rate controllers for shaping QoS traffic can also yield delay bounds for the QoS traffic. We will now describe a queueing architecture to provide delay bounds in a crossbar switch.

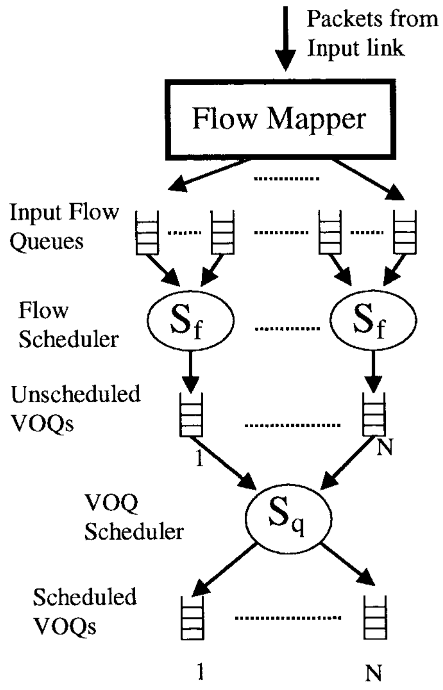


Fig. 3. Queueing architecture at the input to support QoS traffic.

A. A Queueing Architecture to Support QoS Traffic

It is implicitly assumed that there is a rate associated with each flow. The service parameters of each flow are communicated to the switch during connection establishment. It is also assumed that the flow loads are feasible. Please refer to Fig. 3 for the following discussion. Each input channel maintains a per-flow queue. Packets upon arrival from the physical input link are mapped to a flow through a flow mapper. The flow can be based on various classifiers: source address, destination address, protocol type, etc. The packet is stored in the memory, and a pointer to that packet is added to the queue corresponding to the flow. There is a (police) rate controller S_f, one per VOQ, that schedules flows into their respective VOQ's. When a flow is scheduled, the pointer to the HOL cell in the selected flow's queue is removed and added to the tail of the VOQ corresponding to the flow. However, since we have one S_f per VOQ, there can be multiple cells simultaneously entering different VOQ's per cell time. In order to maintain one-cell arrival[2] per cell time, we employ another rate controller S_q that schedules a single VOQ per cell time. Thus, we have two types of VOQ's[3]—*unscheduled* and *scheduled.* When the rate controller S_q schedules a VOQ, the pointer to the HOL cell in the selected *unscheduled* VOQ is removed and added to the tail of the *scheduled* VOQ. The matching algorithm *only* considers cells from the *scheduled* VOQ's. It is important to note that in the discussion in the previous sections, the arbiter considered the HOL cells in all VOQ's. It was essential for the work-conserving property that no more than one cell could enter the input. In the framework we discuss here, the arbiter considers only the cells which "arrive" to it from the two-level rate controller, and so the property of one "arrival" per cell slot holds with respect to such cells. The idea here is to "pretreat" traffic arriving to the input so that by the time the arbiter "sees" the traffic, the combined traffic at all inputs destined to a particular output conforms closely to the output rate [15], [20]. As will be shown in the next section, the combination of the use of rate controllers and the work-conserving property of LOOFA provide delay guarantees for the *scheduled* cells.

[2]Limiting to one-cell arrival per cell slot is essential for the correct operation of LOOFA.

[3]Of course, we need not have two separate queues to implement it. We could have just maintained a bit with each cell and set the bit when it gets scheduled. We explain it this way purely for the sake of clarity.

B. Arbitration Delay Bounds

The end-to-end switch delay of a cell D is the sum of the delay incurred at the input, the arbitration delay (D_a), and the delay incurred at the output (D_o). The input delays are due to the rate controllers and are not dependent on the matching algorithm used. In this paper, we will derive the arbitration and the output delays due to LOOFA. The key delay component in crossbar switches is the arbitration delay, which critically depends on the matching algorithm used. The arbitration delay of a cell is the time taken to reach the output channel after it is scheduled. We prove in the Appendix that using the best-first version of LOOFA and employing OCF as the input selection (best-first LOOFA-OCF) algorithm for $S \geq 2$, $D_a \leq 4N/(S-1)$ cell slots. This delay bound is for the case when the input rate controllers S_f and S_q employ a rate-controlled version of WF^{2}Q.[4] It is also shown that if the output is arranged in a FIFO manner, $D_o \leq 2N$ cell slots. Thus, for $S=2$, the bound on the sum of the arbitration and output delay is $6N$ cell slots. This delay bound did not require the switch to emulate an output-buffered switch.

In [14], it has been proven that best-first LOOFA-OCF emulates an output-buffered switch (with the output arranged in a FIFO) for a speedup of only three. It is shown in the Appendix that by using the queueing architecture presented in the previous section at the input, we can obtain a bound of only $2N$ cell slots for $(D_a + D_o)$ for a switch which emulates an output-buffered switch (with the output arranged in a FIFO). Thus, the bound for $(D_a + D_o)$ using best-first LOOFA-OCF for $S=2$ is slightly worse than $S=3$.

VI. Complexity Comparison

In this section we will compare the complexity of best-first LOOFA-OCF with the *critical cell first* (CCF) algorithm that exactly emulates output-buffered switches [19]. There are two primary operations that lend to the complexity of an iterative crossbar maximal-matching algorithm:

a) assigning/updating priority of the cells;
b) computation of the maximal match.

The priority determines the order in which cells are selected to send a request from an input and also the order in which

[4]The delay bounds depend on the discrepancy bounds of the rate controller. WF^{2}Q was chosen because it is known to have a very small discrepancy bound and therefore can ensure small delays [13].

requests are selected at an output. In the case of LOOFA-OCF, the priority of a cell is a function of the OCC and TS. The input ordering is determined by the OCC of the output of the cell, and the output ordering is determined by the TS of the cell. If the TS is the time of arrival into the VOQ (which is the case in LOOFA-OCF), it is invariant during the lifetime of the cell in the switch. However, the occupancy of an output can change over a cell slot, thus causing a change in the priority of the cells (those that are still at the input side of the switch) going to the output. For example, if the occupancy of an output increases during a cell slot, the priority of the cells going to the output has to decrease. Thus, at the end of each cell slot, there needs to be a mechanism to update the priorities of the cells residing at the inputs. Since the occupancies are stored in the arbiter (see Section IV), they can be centrally updated. Moreover, only at most N occupancy values get updated (one per output port) instead of updating values for each cell in the switch. Typical range of N for WAN switches is 8–64. Thus, at the beginning of each phase, the inputs just need to send the TS of the HOL cells of its active VOQ's. Along with the locally stored occupancy values, the arbiter has all the information it requires to run the matching algorithm.

In order to exactly emulate output-buffered switches, cells in the CIOB switch have to exit the switch at exactly the same time as in the shadow output-buffered switch [19]. Let the time of departure of cell c from the shadow switch be $\mathrm{TL}(c)$. We need to keep track of the "output cushion" $(\mathrm{OC}(c))$ of each cell c, which is the number of cells that are currently at the output and have a lower TL than cell c. Mere output occupancy tracking is not enough.[5] The priority of a cell c is its output cushion, which changes every cell slot. Unlike occupancy, all cells going to a particular output do not have the same priority. In the worst case, at the end of each cell slot, the priorities of all the cells at the inputs have to be updated. The priority information of the cells can either be maintained at the arbiter (thus increasing the storage overhead) or maintained at the input (thus increasing the communication overhead during requests and updates). One may argue that since the relative priorities of the cells at the input do not change, we need not update the priorities. This is not true because when a new cell arrives, it has to be placed in the correct position in the ordered list at the input for the algorithm to perform correctly. More specifically, a new cell that arrives into an empty VOQ will need to know its priority at the time of its arrival because it can take part in the matching process in that cell slot itself. Thus, the update overhead in the case of CCF is proportional to the number of cells in the switch, which can be extremely high.

The complexity due to the computation of the maximal match in best-first LOOFA-OCF is similar to that in CCF. Both require N (switch size) iterations for correct operation. By correct operation in the context of LOOFA, we mean that work conservation and delay bounds are not violated. In the context of CCF, correct operation means exact emulation of the output-buffered switch.

[5]Actually, exact emulation is possible using occupancy. However, it requires a speedup of three [14].

VII. Conclusion

There is an immediate demand for high capacity switches and routers that can provide both work-conserving properties and also rate and delay guarantees. Presented in this paper was a novel crossbar-arbitration algorithm, LOOFA, which is work conserving for all traffic patterns and switch sizes for a speedup of only two. Also presented were a number of schemes that can be used in combination with LOOFA to provide a wide range of delay and fairness properties. For a rate-controlled input, and using best-first LOOFA-OCF with a speedup of two, bounds were derived for the arbitration and output queueing delays. We also show that these delay bounds are achievable without the need to emulate an output-buffered switch. To the best of our knowledge, LOOFA is the only arbitration algorithm that is easy to implement and provides both work conservation and delay guarantees in a crossbar for a speedup of only two.

Appendix
Proof for the Work-Conserving Property

The following terminology is used in the claims and proofs.

C_{ij}	A cell going from input i to output j.
S	Speedup of the switch fabric.
$X_j(t)$	The number of cells in output j at time t.
$IT(C_{ij}(t))$	For a cell C_{ij} going from input i to output j, it is the set of cells C_{ij} (where $1 \le j' \le N$) that have occupancy $X_{j'}(t) \le X_j(t)$ at t, except itself.
$\lvert IT(C_{ij}(t))\rvert$	Size of the input thread of cell C_{ij} at t.
$NI(C_{ij}(t))$	Number of cells sent from a cell C_{ij}'s input thread in tth cell slot.
$NO_j(t)$	Number of cells sent to output j in tth cell slot.
$C(t)$	Set of all the cells at the input of the switch at t.
t_b (t_e)	Beginning (end) of the tth cell slot.
$New(t_b)$	Set of cells that arrived into the switch at the beginning of cell slot t.
$Xfer(t_e)$	Set of cells that were transferred to the output of the switch at the end of the cell slot t.

The following notations are specific to the *proofs for delay bounds*.

$TS(C_{ij})$	Timestamp of the cell C_{ij}. In the best-effort LOOFA-OCF scheme, the timestamp of a cell is the time the cell was scheduled by S_q.
r_{ij}	Sum of the rates of all flows going from input i to output j.
$OT(C_{ij}(t))$	For a cell C_{ij}, it is the set of cells $C_{i'j}$ (where $1 \le i' \le N$) that have timestamps less than or equal to the timestamp of C_{ij}.
$\lvert OT(C_{ij}(t))\rvert$	Size of the output thread of cell C_{ij} at t.
$A_j^i(t)$	Number of cells scheduled for output j from input i by the end of cell slot t.
$A_j(t)$	$= \sum_{i=1}^{N} A_j^i(t)$. Number of cells scheduled for output j by the end of cell slot t.
$D_j(t)$	Number of scheduled cells that exit output j by the end of cell slot t.

N_j^o	Number of scheduled cells for output j that remains in the switch by the end of cell slot t. It is basically equal to $A_j(t) - D_j(t)$.
$N_i^I(t)$	Number of scheduled cells at input i by the end of cell slot t.

Lemma 1: Consider a switch running LOOFA (greedy or best-first version). Consider a tagged cell C_{ij} that at the beginning of phase P is at the input of the switch. If C_{ij} remains in its input at the end of phase P and is not forwarded to its output, at least one cell was transferred from its input thread, and/or one cell was transferred to its output during the phase P.

Proof: In the best-first version of LOOFA only one output (specifically, the lowest-occupancy unmatched output across the switch) gets matched every iteration. The selected output gets matched to the unmatched input that has the cell destined to it with the lowest timestamp (in the case of OCF-based input selection). Under the operation of such an algorithm, if a cell did not go during a phase, then a different cell must have left its input thread and/or its output thread. In the greedy version of LOOFA, the inputs initiates the matching operation. Each input selects the lowest unmatched output amongst the outputs for which it has a cell and sends a request to it. Multiple outputs can get requests from inputs. Each requested output selects the input with the lowest timestamp (in the case of OCF). Under the operation of such an algorithm, if a cell does not go during a phase, then a different cell must have left its input thread and/or gone to its output. Unlike the best-first version, if a cell did not leave its input thread, it is not guaranteed that a cell from its output thread would have left. However, for work conservation, it suffices that Lemma 1 is true—which happens to be the case for both the greedy and the best-first version of LOOFA. □

Corollary 1: Consider a switch running the lowest-occupancy first algorithm with a speedup of S. Consider a tagged cell C_{ij} destined for output j, which at the beginning of cell slot T (T_b) is in the input i of the switch. If C_{ij} remains in its input at the end of cell slot T (T_e) and is not forwarded to its output j, then $NI(C_{ij}(T)) + NO_j(T) \geq S$.

Proof: Follows from Lemma 1. □

Lemma 2: At an input, the length of the input thread of all the cells going to outputs having the same occupancy value are equal.

A. Proof for Work Conservation

For the purpose of the proof, we use the following occupancy update procedure. At the end of a cell slot T (T_e), the occupancy of the output j, called X_j, is decremented by one (simulating a cell departure to its outgoing link) only if there was a cell in the output FIFO at the beginning of the cell slot T (T_b) or if there was at least one cell at any input of the switch destined for this output at T_b. Thus, in the presence of such an updating procedure for X_j, the instant X_j for any output j of the switch becomes less than zero, the switch will cease to be work conserving. In Theorem 1, we will prove a stronger invariant: for each and every cell (say, C_{ij}) in the switch operating under LOOFA with a speedup of two, during its tenure at the input of the switch, $X_j(t_e) \geq |IT(C_{ij}(t_e))|$. Therefore, the occupancy of all outputs in a switch operating under LOOFA and a speedup of two is always nonnegative.

Theorem 1: Consider an $N \times N$ switch operating under LOOFA with a speedup of S. Suppose that the switch has been operating from cell slot 1, having been empty before that time. For each cell C_{ij} in the switch, as long as the cell remains at the input, $X_j(t_e) \geq |IT(C_{ij}(t_e))|$ holds, so long as $S \geq 2$ holds. $T_{\text{arr}_e} \leq t_e \leq T_{\text{xfer}_e}$ where T_{arr_e} is the end of the cell slot T_{arr} (the cell C_{ij} arrives at T_{arr_b}), and T_{xfer_e} is the end of the cell slot T_{xfer} (the cell C_{ij} gets transferred to its output during T_{xfer}).

Proof: New arrivals for the tth cell slot take place at t_b. For a speedup $S = 2$ over a cell slot, the occupancy of the outputs that has at least one cell in the input changes in one of the following ways:

Case 1) $X_j(t_e) = X_j((t-1)_e - 1$; Let $J^1(t_e) = \{j : X_j(t_e) = X_j((t-1)_e) - 1\}$.

Case 2) $X_j(t_e) = X_j((t-1)_e$; Let $J^2(t_e) = \{j : X_j(t_e) = X_j((t-1)_e)\}$.

Case 3) $X_j(t_e) = X_j((t-1)_e$; Let $J^3(t_e) = \{j : X_j(t_e) = X_j((t-1)_e) + 1\}$.

Let

$$C^K(t_e) = \{C_{ij} : C_{ij} \in (C((t-1)_e) \cap C(t_e)), j \in J^K(t_e)\}$$
$$New^K(t_e) = \{C_{ij} : C_{ij} \in (New(t_b) \cap C(t_e)), j \in J^K(t_e)\}$$

where $1 \leq K \leq 3$

$$\eta^K(t_e) = C^K(t_e) \cup New^K(t_e), \quad 1 \leq K \leq 3.$$

Thus, $New^K(t_e)$ consists of new cells that arrived at t_b and are still at the input of the switch at t_e, and the outputs of the cells are in class K at t_e. It follows that

$$N = \sum_{K=1}^{3} |J^K(t_e)| \quad \text{and}$$
$$C(t_e) = \bigcup_{1 \leq K \leq 3} (C^K(t_e) \cup New^K(t_e)).$$

Given that there can at most be one new arrival into an input per cell slot, $\forall C_{ij} \in C(t_e), \forall L_{ij} \in C((t-1)_e)$

$$|IT(C_{ij}(t_b))| \leq |IT(L_{ij}((t-1)_e)| + 1. \qquad (1)$$

Let us assume there were no violations until the end of the $(t-1)$th cell slot. Thus

$$\forall L_{ij} \in C((t-1)_e), \quad X_j((t-1)_e) \geq |IT(L_{ij}((t-1)_e))|. \qquad \text{(A1)}$$

It follows from (1) and (A1) that

$$\forall C_{ij} \in C(t_e), \quad |IT(C_{ij}(t_b))| \leq X_j((t-1)_e) + 1. \qquad (2)$$

We will now show that there is no violation at t_e,i.e., $\forall C_{ij} \in C(t_e), X_j(t_e) \geq |IT(C_{ij}(t_e))|$ holds.

Case 1: As per Assumption (A1), $\forall L_{ij} \in C^1(t_e), X_j((t-1)_e) \geq |IT(L_{ij}((t-1)_e))|$. Since no cells were sent to output $j \in J^1(t_e)$, two cells must have gone from the input thread of the cells $C_{ij} \in \eta^1(t_e)$ (from Corollary 1). Now we have to determine the increase in the input thread. An increase in the input thread can happen due to a new cell arrival and/or other cells that at $(t-1)_e$ had an occupancy more than $X_j(t_e)$ and at t_e have an occupancy less than or equal to $X_j(t_e)$. However, since we know that occupancy can decrement by at most one, outputs with occupancy value more than output j at $(t-1)_e$ cannot have a lower occupancy than j at t_e. Thus, for all the cells in Case 1, the input thread can increase *only* due to a new cell arrival. Due to the new cell arrival at t_b, at most one new cell can become part of its input thread. Thus, from (2) $\forall C_{ij} \in \eta^1(t_e), \forall j \in J^1(t_e)$

$$\begin{aligned}|IT(C_{ij}(t_e)| &\leq |IT(C_{ij}(t)_b)| - 2 \leq X_j(t-1)_e + 1 - 2\\ &\leq X_j(t-1)_e - 1.\end{aligned}$$

We know that for all the cells in Case 1, $X_j(t_e) = X_j((t-1)_e - 1$.

Thus, $\forall C_{ij} \in \eta^1(t_e)$, there is no violation. CLAIM 1

Case 2: As per Assumption (A1), $\forall C_{ij} \in C^2(t_e), X_j((t-1)_e) \geq |IT(C_{ij}((t-1)_e)|$. Let us assume that there is a violation at t_e, i.e.,

$$\{\exists C_{ij} : |IT(C_{ij}(t_e))| > X_j(t_e), C_{ij} \in \eta^2(t_e)\}. \quad \text{(A2)}$$

For the outputs $j \in J^2(t_e)$, exactly one cell was sent to j in the cell slot t. For a speedup $S = 2$ (from Corollary 1) at least one cell must have gone from the cell C_{ij}'s (where $C_{ij} \in \eta^2(t_e)$) input thread. Let the increase in input thread due to other cells in the input be δ. Thus, from (1)

$$\begin{aligned}&\forall C_{ij} \in \eta^2(t_e), \quad \forall L_{ij} \in C((t-1)_e), \quad \forall j \in J^2(t_e),\\ &\quad |IT(C_{ij}(t_e))| \leq |IT(L_{ij}((t-1)_e)| + 1 - 1 + \delta\\ &\qquad \leq |IT(L_{ij}((t-1)_e)| + \delta\\ &\forall C_{ij} \in \eta^2(t_e), \quad X_j(t_e) = X_j((t-1)_e.\end{aligned}$$

Since the occupancy value of the cell C_{ij} ($C_{ij} \in \eta^2(t_e)$) does not change in this time slot, and since occupancy can decrease by at most one, only cells that can become new additions to C_{ij}'s input thread are the cells C_{ik} for which the occupancy value $X_k((t-1)_e = X_j(t_e) + 1$, and the occupancy of these cells decreased by one over the cell slot t. Thus, $C_{ik} \in \eta^1(t_e)$. Since the occupancy value of these cells C_{ik} is equal to C_{ij} at t_e, the length of the input thread at t_e are equal too (from Lemma 2). Thus, as per Assumption (A2), if there is a violation for a cell $C_{ij} \in \eta^2(t_e)$, then there should have been a violation for a cell $C_{ik} \in \eta^1(t_e)$. This however contradicts Claim 1. Thus, there are no violations for $\forall C_{ij} \in \eta^2(t_e)$. CLAIM 2

Case 3: As per Assumption (A1), $\forall C_{ij} \in C^3(t_e), X_j((t-1)_e) \geq |IT(C_{ij}((t-1)_e)|$. Let us assume that there is a violation at t_e, i.e.,

$$\{\exists C_{ij} : |IT(C_{ij}(t_e))| > X_j(t_e), C_{ij} \in \eta^3(t_e)\}. \quad \text{(A3)}$$

For the outputs $j \in J^3(t_e)$ exactly two cells were sent to j in the cell slot t. For a speedup $S = 2$, from Corollary 1, no cells could have gone from the cell C_{ij}'s (where $C_{ij} \in \eta^3(t_e)$) input thread during t. Let the increase in input thread due to other cells in the input be δ. Thus, from (1)

$$\forall C_{ij} \in \eta^3(t_e), \quad |IT(C_{ij}(t_e))| \leq |IT(C_{ij}((t-1)_e)| + 1 + \delta.$$

We know for cells in Case 3, $X_j(t_e) = X_j((t-1)_e + 1$. For Assumption (A3) to be valid, $|IT(C_{ij}(t_e))| > X_j(t_e)$ holds true for some cell $C_{ij} \in \eta^3(t_e)$. Only cells that can become new additions to C_{ij}'s input thread at t_e are $C_{ik} \in \eta^1(t_e) \cup \eta^2(t_e)$. These are comprised of cells C_{ik}'s in $\eta^1(t_e)$ that had occupancy $X_k((t-1)_e) = X_j(t_e) + 2$ and cells C_{ik}'s in $\eta^2(t_e)$ that had occupancy $X_k((t-1)_e) = X_j(t_e) + 1$. At $t_e, X_k(t_e) = X_j(t_e)$. Since the occupancy values are equal, the lengths of the input threads are equal too (as per Lemma 2). Thus, as per Assumption (A3), if there was a violation for a $C_{ij} \in \eta^3(t_e)$ at t_e, then there should have been a violation for $C_{ik} \in \eta^1(t_e) \cup \eta^2(t_e)$. This however contradicts Claims 1 and 2. Hence, there must have been no violations for $C_{ij} \in \eta^3(t_e)$. CLAIM 3

Thus, from Claims 1, 2, and 3, $\forall C_{ij} \in C(t_e), X_j(t_e) \geq |IT(C_{ij}(t_e))|$. □

Theorem 2: An $N \times N$ switch running the lowest-occupancy first algorithm is work conserving regardless of the input traffic patterns and for arbitrary values of N, so long as its speedup $S \geq 2$.

Proof: Since, for each cell as long as the cell remains at the input $|IT(C_{ij}(t))| \geq 0$, it follows from Theorem 1 that $X_j(t) \geq 0$. Thus, the switch is work conserving. □

B. Proof for Arbitration and Output Delay Bounds

Lemma 3: $\forall j, t, N_j^o(t) \leq 2N$ holds true in an $N \times N$ switch running best-first LOOFA-OCF algorithm for WF^{2}Q-based rate controlled inputs, as long as $S \geq 2$.

Proof: From [13], we know that following holds for WF^{2}Q based rate controllers: $r_{ij}t - 1 \leq A_j'(t) \leq r_{ij}t + 1$. Thus

$$\sum_{i=1}^{N}(r_{ij}t - 1) \leq A_j(t) \leq \sum_{i=1}^{N}(r_{ij}t + 1)$$
$$L_jt - N \leq A_j(t) \leq L_jt + N$$

where L_j is the total load at the output j. Now, since LOOFA is work conserving as long as $S \geq 2$, the number of cells exiting an output should be at least equal to the lower bound on the number of cells scheduled for that output. Thus, $D_j(t) \geq L_jt - N$. Thus, the maximum number of scheduled cells for an output j that can remain in the switch at any time is

$N_j^o(t) \leq (L_jt + N) - (L_jt - N) \leq 2N$. □

Corollary 2: $\forall j, t, X_j(t) \leq 2N$ and $\forall C_{ij} \in C(t), t, |OT(C_{ij}(t))| \leq 2N$ holds true in an $N \times N$ switch running the best-first LOOFA-OCF algorithm for WF^{2}Q-based rate controlled inputs, as long as $S \geq 2$.

Proof: Since all cells scheduled for an output either reside at the input side of the switch and the output FIFO, it follows from Lemma 3. It also follows that the delay any cell encounters at the output D_o is bounded by $2N$. □

Corollary 3: $\forall C_{ij} \in C(t), t, |IT(C_{ij}(t))| \leq 2N$ holds true in an $N \times N$ switch running the best-first LOOFA-OCF algorithm for WF^{2}Q-based rate controlled inputs, as long as $S \geq 2$. It also follows that $\forall i, t, N_i^I(t) \leq 2N + 1$ also holds.

Proof: Follows from Corollary 2 and Theorem 1. □

Theorem 3: The arbitration delay D_a of an $N \times N$ switch running the best-first LOOFA-OCF algorithm for WF^{2}Q-based rate controlled inputs is bounded by $4N/(S-1)$, as long as $S \geq 2$.

Proof: The only cells that a cell competes with are the cells in its output thread and its input thread. Thus, any new cell arriving into the switch competes with at most $4N$ cells ($2N$ due to its input thread and $2N$ due to its output thread) (follows from Corollaries 2 and 3). From Lemma 1, it follows that, using best-first LOOFA-OCF, in each phase, if a tagged cell C_{ij} does not go, then a cell from its output thread or/and *input thread* goes. It also implies that if a cell C_{ij} does not go, a cell from its output thread or/and its *input* goes. Thus, during each phase, the sum $(N_i^I(t) + OT(C_{ij}(t))$ decreases at least by one. $N_i^I(t)$ can increase by one during each cell slot due to a new cell arrival. Thus, under a speedup of S, the sum $(N_i^I(t) + OT(C_{ij}(t))$ decreases at least by $S - 1$. □

Thus, at the end of $4N/(S-1)$ cell slots, if the tagged cell is still at the input, there will be no cells competing with it—in other words, there will be no cells in its output thread nor in its input (apart from itself). Thus, in the next cell slot, only the new cell that comes into its input competes with it, and hence, the tagged cell will go to its output in one of the S phases. Thus, $D_a \leq 4N/(S-1)$.

Corollary 4: The delay $(D_a + D_o)$ of an $N \times N$ output-buffered switch employing a FIFO policy at the output for WF^{2}Q-based rate controlled inputs is bounded by $2N$.

Proof: $D_a + D_o$ is the delay incurred by the cell from the time it exits the rate controller and the time it exits the switch output. From Lemma 3, a cell exiting the rate controller notices at most $2N$ cells ahead of it at the output in a output-buffered switch. Since the output is serviced in a FIFO manner, $(D_a + D_o)$ is bounded by $2N$. □

ACKNOWLEDGMENT

The authors are grateful to J. Saxe and T. Rodeheffer of the Systems Research Center at Compaq Computer Corporation for verifying the work-conservation proofs.

REFERENCES

[1] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space division switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347–1356, 1987.

[2] S.-Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," in *Proc. IEEE Globecom*, 1989, pp. 1754–1763.

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

[4] R. E. Tarjan, "Data Structures and Network Algorithms," Bell Labs, 1983.

[5] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, pp. 319–352, Nov. 1993.

[6] D. Stiliadis and A. Verma, "Providing bandwidth guarantees in an input-buffered crossbar switch," in *Proc. IEEE INFOCOM*, 1995, pp. 960–968.

[7] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, Univ. California, Berkeley, CA, 1995.

[8] C. Lund, S. Phillips, and N. Reingold, "Fair prioritized scheduling in an input-buffered switch," in *Proc. Broadband Commun.*, 1996.

[9] N. McKeown, B. Prabhakar, and M. Zhu, "Matching output queueing with combined input and output queueing," in *Proc. 35th Ann. Allerton Conf. Commun., Control, and Computing*, Oct. 1997.

[10] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," Computer Systems Lab, Stanford Univ., Stanford, CA, Tech. Rep. CSL-TR-97-738.

[11] R. Guerin and K. N. Sivarajan, "Delay and throughput performance of speeded-up input-queueing packet switches," IBM, Res. Rep. RC 20892, 1997.

[12] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. IEEE INFOCOM*, Mar. 1998.

[13] J. Bennett and H. Zhang, "WF^{2}Q—Worst-case fair weighted fair queueing," in *Proc. INFOCOM*, 1996.

[14] T. L. Rodeheffer and J. B. Saxe, "An efficient matching algorithm for a high-throughput, low-latency data switch," Compaq Comput. Corp., Syst. Res. Ctr., Res. Rep. 1998-162.

[15] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbar switches with speedup," in *Proc. IWQoS*, Napa Valley, CA, May 1998.

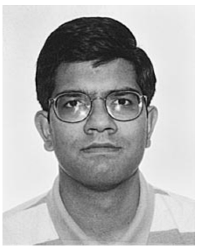
[16] Y.-S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1274–1283, Oct. 1987.

[17] Y. Oie *et al.*, "Effect of speedup in nonblocking packet switch," in *Proc. ICC*, 1989.

[18] J. S.-C. Chen and T. E. Stern, "Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 439–449, Apr. 1991.

[19] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," Stanford Univ., Stanford, CA, Tech. Rep. CSL-TR-98-758. (An extended version of this report also appears in this special issue.)

[20] A. Charny, "Providing QoS guarantees in input-buffered crossbar switches with speedup," Ph.D. Dissertation, MIT, Cambridge, MA, 1998.



Pattabhiraman Krishna received the B.S. degree in electrical engineering from Regional Engineering College, Rourkela, India, in 1990, the M.S. degree in electrical engineering from Texas A&M University, College Station, TX, in 1992, and the Ph.D. Degree in computer science, also from Texas A&M University, in 1996.

Since August 1996, he has been a member of the Advanced Development Group with Networks Products Business, Compaq Computer Corp., (formerly Digital Equipment Corp.), Littleton, MA. His research interests are in the areas of data networking and mobile computing.

Naimish S. Patel received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1996.

From 1996 to 1997, he was a staff member at Lincoln Laboratories, Lexington, MA. In March 1997, he joined the Network Products Group, Digital Computer Corp., Littleton, MA. In February 1998, he joined Sycamore Networks, Chelmsford, MA, as a Founding Engineer, where he currently is Network Engineer and Lead Designer of DWDM systems.

Anna Charny received the M.S. degree in mathematics from Kalinin State University, Russia, and the M.S. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge.

Until 1998, she was with the Networks and Communications Division, Digital Equipment Corp., Littleton, MA. Currently, she is with the Digital Network Products Group, Cabletron Corp., Andover, MA.

Dr. Charny has served as Technical Expert for the Traffic Management Group of the ATM Forum. She is a Member of the Editorial Board for IEEE NETWORKING MAGAZINE. Her research interests are performance analysis and design of algorithms for high-speed communications networks.

Robert J. Simcoe was born in Columbus, OH, on November 5, 1943. He received the B.S.E.E. degree from the University of Illinois, Urbana, in 1966.

From 1966 to 1973, he worked at the National Security Agency. From 1973 to 1982, he worked at the Integrated Circuit Center, General Electric. From 1982 to 1997, he worked at Digital Equipment Corp., Littleton, MA. He is now the Corporate Consulting Engineer at Nexabit Networks Inc., Marlboro, MA, where he is working on Terabit switching and routing.