Achieving 100% Throughput in an Input-Queued Switch

Nick McKeown, *Senior Member, IEEE,* Adisak Mekkittikul, *Member, IEEE,* Venkat Anantharam, *Fellow, IEEE*, and Jean Walrand, *Fellow, IEEE*

***Abstract*— It is well known that head-of-line blocking limits the throughput of an input-queued switch with first-in–first-out (FIFO) queues. Under certain conditions, the throughput can be shown to be limited to approximately 58.6%. It is also known that if non-FIFO queueing policies are used, the throughput can be increased. However, it has not been previously shown that if a suitable queueing policy and scheduling algorithm are used, then it is possible to achieve 100% throughput for all independent arrival processes. In this paper we prove this to be the case using a simple linear programming argument and quadratic Lyapunov function. In particular, we assume that each input maintains a separate FIFO queue for each output and that the switch is scheduled using a maximum weight bipartite matching algorithm. We introduce two maximum weight matching algorithms: longest queue first (LQF) and oldest cell first (OCF). Both algorithms achieve 100% throughput for all independent arrival processes. LQF favors queues with larger occupancy, ensuring that larger queues will eventually be served. However, we find that LQF can lead to the permanent starvation of short queues. OCF overcomes this limitation by favoring cells with large waiting times.**

***Index Terms*— Arbitration, ATM, input-queued switch, input-queueing, packet switch, queueing networks, scheduling algorithm.**

I. Introduction

SINCE Karol *et al.*'s paper was published in 1986 [11], it has become well known that an $N \times N$ port input-queued switch with first-input–first-output (FIFO) queues can have a throughput limited to just $(2-\sqrt{2}) \approx 58.6\%$. The conditions for this to be true are that:

1) arrivals at each input are independent and identically distributed (i.i.d.);
2) arrival processes at each input are independent of arrivals at other inputs;
3) all arrival processes have the same arrival rate and destinations are uniformly distributed over all outputs;

Paper approved by P. E. Rynes, the Editor for Switching Systems of the IEEE Communications Society. Manuscript received July 29, 1997; revised October 5, 1998. This paper was presented in part at INFOCOM'96, San Francisco, CA.

N. McKeown is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9030 USA (e-mail: nickm@stanford.edu).

A. Mekkittikul was with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9030 USA. He is now with Berkeley Concept Research Corporation, Berkeley, CA 94704 USA (e-mail: adisak@bcrcorp.com).

V. Anantharam is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: ananth@eecs.berkeley.edu).

J. Walrand is with Odyssia Systems, Berkeley, CA 94710 USA (e-mail: jean@odyssiasystems.com).

Publisher Item Identifier S 0090-6778(99)06305-9.

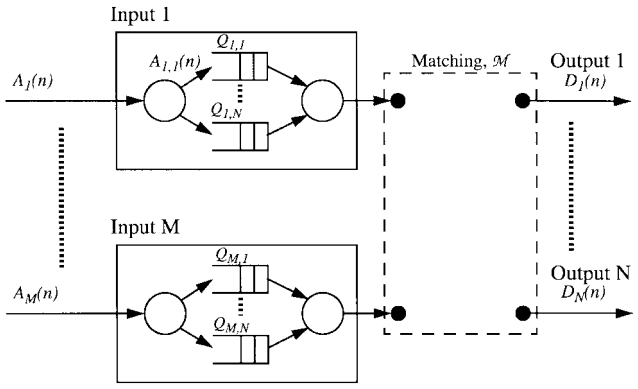


Fig. 1. Components of an input-queued cell-switch.

4) arriving packets are of fixed and equal length, called cells;
5) N is large.

When conditions 1) and 2) are true we shall say that arrivals are *independent*, and when condition 3) is true we shall say that arrivals are *uniform*.

The throughput is limited because a cell can be held up by another cell ahead of it in line that is destined for a different output. This phenomenon is known as head-of-line (HOL) blocking.

It is well documented that this result applies only to input-queued switches *with FIFO queues*. And so many techniques have been suggested for reducing HOL blocking using non-FIFO queues, for example, by examining the first K cells in a FIFO queue, where $K > 1$ [5], [8], [10]. In fact, HOL blocking can be eliminated entirely by using a simple buffering strategy at each input port. Rather than maintain a single FIFO queue for all cells, each input maintains a separate queue for each output [1], [9], [16]–[19], as shown in Fig. 1. This queuing discipline is often referred to as virtual output queuing (VOQ). HOL blocking is eliminated because a cell cannot be held up by a cell queued ahead of it that is destined for a different output. The implementation of VOQ is slightly more complex, requiring N FIFO's to be maintained by each input buffer. But no additional speedup is required; at most one cell can arrive and depart from each input in a slot. During each slot, a scheduling algorithm decides the configuration of the switch by finding a matching on a bipartite graph (described below). A number of different techniques have been used for finding such a matching, for example, using neural networks [2], [4], [22] or iterative algorithms [1], [14], [15]. These algo-

rithms were designed to give high throughput while remaining simple to implement in hardware. When traffic is uniform, these algorithms perform well (>90% throughput). The iSLIP algorithm [14], [15], for example, has been demonstrated using simulation to achieve 100% throughput when the traffic is independent and uniform. However, all of these algorithms perform less well and are unable to sustain a throughput of 100% when traffic is nonuniform.

It is worth asking the question:

> What is the highest throughput that can be achieved by an input-queued switch which uses the queueing discipline shown in Fig. 1?

In this paper we prove that for independent arrivals (uniform or nonuniform), a maximum throughput of 100% is achievable using two maximum weight matching algorithms.

In Section II we describe our model for an input-queued switch that uses virtual output queuing, as illustrated in Fig. 1. We then consider three graph algorithms that can be used to schedule the transfer of cells through the switch. First, in Section III, we describe the "maximum size" algorithm. Although this algorithm achieves 100% throughput for uniform traffic, we show that it can become unstable, even starve input queues, when arrivals are nonuniform. Next, in Section IV, we describe two maximum weight scheduling algorithms that overcome this limitation: LQF and OCF. In conjunction with the Appendix, we prove that these two scheduling algorithms are stable for all uniform and nonuniform independent arrival processes up to a maximum throughput of 100%. It is important to note that this is a theoretical result—the maximum weight matching algorithms that we propose are not readily implemented in hardware. Furthermore, it should be noted that the aim of this paper is to find algorithms that achieve 100% throughput for best-effort traffic. No attempt is made to achieve fairness between different flows, or to provide guaranteed qualities of service. We expect future results will combine algorithms that achieve high throughput (such as those presented here) with additional mechanisms that impose fairness or bandwidth guarantees (e.g., round-robin or weighted round-robin). Our result indicates that practical techniques approximating the algorithms presented here can be expected to achieve high throughput.

II. Our Model

Consider the input-queued cell switch in Fig. 1 connecting M inputs to N outputs. The stationary and ergodic arrival process $A_i(n)$ at input i, $1 \le i \le M$, is a discrete-time process of fixed size packets, or cells. At the beginning of each slot, either zero or one cell arrives at each input. Each cell contains an identifier that indicates which output j, $1 \le j \le N$, it is destined for. When a cell destined for output j arrives at input i, it is placed in the FIFO queue $Q_{i,j}$ which has occupancy $L_{i,j}(n)$. We refer to $Q_{i,j}$ as a VOQ. Define the following vector which represents the occupancy of all queues at slot n

$$\underline{L}(n) \equiv (L_{1,1}(n), \cdots, L_{1,N}(n), \cdots, L_{M,N}(n))^T. \quad (1)$$

Similarly, we define the waiting time $W_{i,j}(n)$ to be the number of time slots spent in the queue by the cell at the head of VOQ $Q_{i,j}$ at time slot n. And we define the following vector to represents the waiting time of the head-of-line cells at all VOQ's at slot n:

$$\underline{W}(n) \equiv (W_{1,1}(n), \cdots, W_{1,N}(n), \cdots, W_{M,N}(n))^T. \quad (2)$$

We shall define the arrival process $A_{i,j}(n)$ to be the process of arrivals at input i for output j at rate $\lambda_{i,j}$, and the set of all arrival processes $A(n) = \{A_i(n); 1 \le i \le M\}$. $A(n)$ is considered *admissible* if no input or output is oversubscribed, i.e., $\sum_i \lambda_{i,j} < 1$, $\sum_j \lambda_{i,j} < 1$, otherwise it is *inadmissible*.

The FIFO queues are served as follows. A scheduling algorithm selects a *match*, or *matching*, $\mathcal{M}$, between the inputs and outputs, defined as a collection of edges from the set of nonempty input queues to the set of outputs such that each nonempty input is connected to at most one output, and each nonempty output is connected to at most one input. At the end of the slot, if input i is connected to output j, one cell is removed from $Q_{i,j}$ and sent to output j. Clearly, the departure process from output j, $D_j(n)$, rate μ_j, is also a discrete-time process with either zero or one cell departing from each output at the end of each slot. We shall define the departure process $D_{i,j}(n)$, rate $\mu_{i,j}$, as the process of departures from output j that were received from input i. Note that the departure rate may not be defined if the departure process is not stationary and ergodic.

To find a matching $\mathcal{M}$, the scheduling algorithm solves a bipartite graph matching problem. An example of a bipartite graph is shown in Fig. 2.

If the queue $Q_{i,j}$ is nonempty, then $L_{i,j}(n) > 0$ and there is an edge in the graph G between input i and output j. We associate a weight $W_{i,j}(n)$ with each such edge. The meaning of the weights depend on the algorithm, and we consider two classes of algorithm here.

Maximum Size Matching Algorithms: Algorithms that find the match containing the maximum number of edges.

Maximum Weight Matching Algorithms: Algorithms that find the maximum weight matching where, in this paper, we consider only algorithms for which the weight $w_{i,j}(n)$ is integer-valued, equaling the occupancy $L_{i,j}(n)$ of $Q_{i,j}$ or the waiting time $W_{i,j}(n)$ of the cell at the head of line at $Q_{i,j}$.

Clearly, a maximum size matching is a special case of the maximum weight matching with weight $w_{i,j}(n) = 1$ when $Q_{i,j}$ is nonempty.

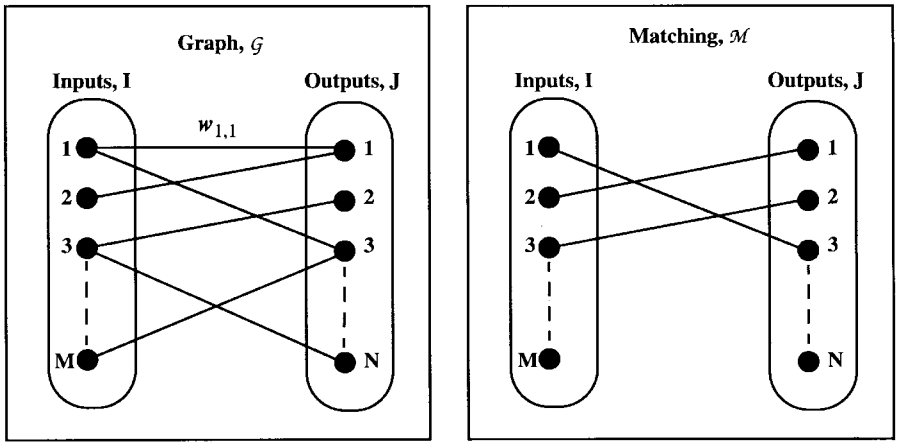
III. Maximum Size Matchings

The maximum size matching for a bipartite graph can be found by solving an equivalent network flow problem [20]. There exist many algorithms for solving these problems, the most efficient algorithm currently known converges in $O(N^{5/2})$ time and is described in [7].[1]

It can be demonstrated[2] using simulation that the maximum size matching algorithm is stable for i.i.d. arrivals up to an offered load of 100% when the traffic is uniform [15]. It is important to note that a maximum size matching is not

[1] This algorithm is equivalent to Dinic's algorithm [6].

[2] Clearly, such a demonstration by simulation is not as strong as an analytical proof.



(a) (b)

Fig. 2. Define $\mathcal{G} = [V, E]$ as an undirected graph connecting the set of vertices V with the set of edges E. The edge connecting vertices i, $1 \leq i \leq M$ and j, $1 \leq j \leq N$ has an associated weight denoted $w_{i,j}$. Graph $\mathcal{G}$ is bipartite if the set of inputs $I = \{i: 1 \leq i \leq M\}$ and outputs $J = \{j: 1 \leq j \leq N\}$ partition V such that every edge has one end in I and one end in J. Matching $\mathcal{M}$ on $\mathcal{G}$ is any subset of E such that no two edges in $\mathcal{M}$ have a common vertex. A *maximum matching algorithm* is one that finds the matching $\mathcal{M}_{\max}$ with the maximum total size or total weight. (a) Example of $\mathcal{G}$ for $|I| = M$ and $|I| = N$. (b) Example of matching $\mathcal{M}$ on $\mathcal{G}$.

necessarily desirable. First, under *admissible* traffic, it can lead to instability and unfairness, particularly for nonuniform traffic patterns. To demonstrate this behavior, Fig. 3 shows an example of a potentially unstable 3×3 switch with just four active flows,[3] and scheduled using the maximum size matching algorithm. It is assumed that ties are broken by selecting among alternatives with equal probability. Arrivals to the switch are i.i.d. Bernoulli arrivals and each flow has arrivals at rate $(1/2) - \delta$, where $\delta > 0$. Even though the traffic is admissible, it is straightforward to show that the switch can be unstable for sufficiently small δ. Consider the event that at slot n, both $A_{2,1}(n)$ and $A_{3,2}(n)$ have arrivals with probability $((1/2) - \delta)^2$ and $L_{1,1}(n) > 0$, $L_{1,2}(n) > 0$, in which case input 1 receives service with probability 2/3. Therefore, the total rate at which input 1 receives service is at most

Final Rate

$$\frac{2}{3}\left(\frac{1}{2}-\delta\right)^2 + \left(1-\left(\frac{1}{2}-\delta\right)^2\right) = 1 - \frac{1}{3}\left(\frac{1}{2}-\delta\right)^2.$$

All inputs have people, so pick any 1 out of 3 matchings

one input doesn't have people

But the arrival rate to input 1 is $1 - 2\delta$, so if

Arrival Rate > service rate of 1

$$2\delta < \frac{1}{3}\left(\frac{1}{2}-\delta\right)^2$$

(which holds for $\delta < 0.0358$), then the switch is unstable and the traffic cannot be sustained by the maximum size matching algorithm.

Second, under *inadmissible* traffic, the maximum size matching algorithm can lead to *starvation*. An example of this behavior is shown in Fig. 4 for a 2×2 switch. It is clear that because all three queues are permanently occupied,

[3] It can also be shown that a 2×2 switch with nonuniform traffic can be unstable when scheduled using a maximum size matching algorithm. However, our proof is more complex and is omitted here.

Total 3 matchings in the graph

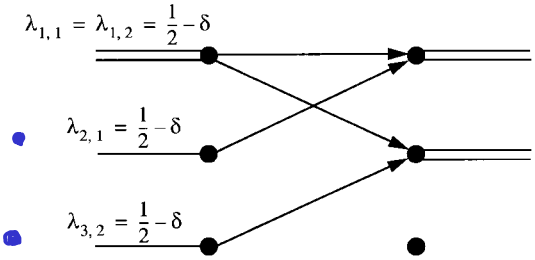


Fig. 3. An example of *instability* under admissible traffic using a maximum size matching algorithm for a 3×3 switch with nonuniform traffic.

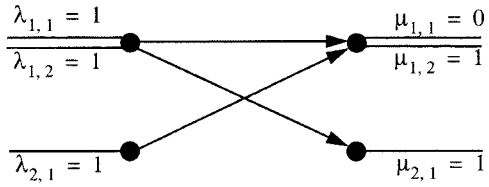


Fig. 4. Under an *inadmissible* workload, the maximum size matching will always serve just two queues, *starving* the flow from input 1 to output 1.

the algorithm will always select the "cross" traffic: input 1 to output 2 and input 2 to output 1. It is worth noting that the practical scheduling algorithms described previously attempt to approximate a maximum size matching [1], [2], [4], [14], [22]. It is therefore not surprising that these algorithms perform well when the traffic is uniform, but perform less well when the traffic is nonuniform.

IV. Maximum Weight Matchings

The maximum weight matching $\mathcal{M}$ for a bipartite graph is one that maximizes $\sum_{(i,j)\in\mathcal{M}} w_{i,j}$ and can be found by solving an equivalent network flow problem. The most efficient known algorithm for solving this problem converges in $O(N^3 \log N)$ running time [20].

The maximum *size* matching algorithm knows only whether an input queue $Q_{i,j}$ is empty or nonempty. Therefore, if the traffic is nonuniform and the occupancy of some queues begins to increase, this algorithm does not know to favor those queues and reduce their backlog.

On the other hand, a maximum *weight* matching algorithm can take into account the occupancy $L_{i,j}(n)$ of each VOQ or the waiting time of the cell at head of line. Such algorithms can give preference to queues with greater occupancy or to older cells. In fact, as our results show, these algorithms can lead to a maximum throughput of 100% for independent and either uniform or nonuniform arrivals.

A. Our Algorithms

In this paper we consider two maximum weight matching algorithms: the "longest queue first" (LQF) algorithm, and the "oldest cell first" (OCF) algorithm. LQF considers the queue occupancy by assigning a weight $w_{i,j}(n) = L_{i,j}(n)$. Queues with larger occupancy will be assigned a larger weight, and are thus more likely to be served.[4] As we shall see, LQF results in 100% throughput. However, LQF can lead to the permanent starvation of a nonempty queue. To understand how this happens, consider a 2×2 switch with $L_{i,j}(0) = 1$ for all i, j, and $\lambda_{1,1} = 1$. In the first timeslot, an arrival will occur at $Q_{1,1}$ and so $Q_{1,2}$ will remain unserved. In fact, because of the continuous arrivals to $Q_{1,1}$, $Q_{1,2}$ will remain unserved indefinitely.

Our second algorithm, OCF, overcomes this problem by considering the waiting times of cells at the head of each VOQ. OCF considers the waiting time by assigning a weight $w_{i,j}(n) = W_{i,j}(n)$. Cells that have been waiting the longest time will be assigned a larger weight and are thus more likely to be served. It is clear that no queues will be starved of service indefinitely: if a cell is not served, its waiting time will increase. Eventually, its weight will increase to a value that ensures that it is served.

V. Main Results

A. The LQF Algorithm

Theorem 1: The LQF maximum weight matching algorithm is stable for all admissible i.i.d. arrival processes.

Proof: The proof is given in Appendix A. In summary, we show that for an $M \times N$ switch scheduled using the LQF algorithm, there is a negative expected single-step drift in the sum of the squares of the occupancy. In other words

$$E\left[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n)|\underline{L}(n)\right] \leq -\varepsilon||\underline{L}(n)|| + k$$

where $k > 0$, $\varepsilon > 0$.

$V(n) = \underline{L}^T(n)\underline{L}(n)$ is a second-order Lyapunov function and, using the result of Kumar and Meyn [13], we show that the system is stable. The term $-\varepsilon||\underline{L}(n)||$ indicates that whenever the occupancy of the input queues is large enough, the expected drift is negative; should $||\underline{L}(n)||$ become very large, the downward drift also becomes large.

[4]When the maximum weight matching algorithm encounters "ties" (i.e., different patterns of connections that lead to different matchings of equal weight), it may arbitrarily select any of the maximum weight matchings. For example, it could randomly choose one matching.

B. The OCF Algorithm

Theorem 2: The OCF maximum weight matching algorithm is stable for all admissible i.i.d. arrival processes.

Proof: The proof is given in Appendix B. The proof consists of two steps. First, we prove the stability of the waiting time. Then, we show that the stability of the waiting time implies the stability of queue occupancy, which proves Theorem 2.

Similar to the LQF proof, we show that for an $M \times N$ switch scheduled using the OCF algorithm, there is a negative expected single-step drift in the value of a second-order Lyapunov function of the waiting times $V(n) = \underline{W}^T(n)T\underline{W}(n)$

$$E\left[\underline{W}^T(n+1)T\underline{W}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\right] \leq -\varepsilon||\underline{W}(n)|| + K$$

where $K > 0$, $\varepsilon > 0$ and T is a positive-definite matrix.

This, in turn, implies the stability of the waiting time.

Once we have proved the stability of the waiting time, it is straightforward to prove the stability of the queue occupancy. Because there can be at most one arrival to any queue in one slot, the total number of arrivals after an HOL cell, by definition the current queue occupancy, is bounded above by the number of slots an HOL cell has been waiting—the waiting time, i.e., $W_{i,j}(n) \geq L_{i,j}(n)\, for all\, i, j, n.$ Therefore, the stability of the waiting time implies the stability of the queue occupancy.

VI. Conclusion

We have shown that if an input-queued switch maintains a separate FIFO queue for each output at each input, then a throughput of 100% can be achieved for independent arrivals. If a maximum-sized matching algorithm is used to schedule cells, then we demonstrate that a throughput of 100% may not be possible when arrivals are nonuniform. However, if a maximum weight matching algorithm is used, we have shown that a throughput of 100% is achievable for both uniform and nonuniform arrivals. In particular, we have described two maximum weight matching algorithms: LQF and OCF. LQF considers the occupancy of the input queues, giving preference to queues that contain more cells. When the occupancy is large enough at any queue, it is ensured of service. Furthermore, when the occupancies of all the queues exceed a threshold, the total queue occupancy exhibits an overall downward drift, ensuring that the total queue occupancy will not become unbounded.

Unfortunately, the LQF algorithm can lead to the indefinite starvation of one or more inputs. We may overcome this limitation by modifying the weights used by the algorithm. In particular, OCF assigns the weights to equal the waiting time of the cell at the head-of-line of each input queue. This

is sufficient to ensure that every cell will eventually be served, and that the system will remain stable.

APPENDIX A
LQF PROOF

A. Definitions

In this appendix we use the following definitions for an $M \times N$ switch.

1) The rate matrix of the stationary arrival processes:

$$\Lambda \equiv [\lambda_{i,j}], \quad \text{where } \sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \lambda_{i,j} \ge 0$$

and associated rate vector

$$\underline{\lambda} \equiv (\lambda_{1,1}, \cdots, \lambda_{1,N}, \cdots, \lambda_{M,1}, \cdots, \lambda_{M,N})^T. \quad (3)$$

2) The arrival matrix, representing the sequence of arrivals into each queue:

$$\mathbf{A}(n) \equiv [A_{i,j}(n)]$$

where

$$A_{i,j}(n) \equiv \begin{cases} 1, & \text{if arrival occurs at } Q(i,j) \text{ at time } n \\ 0, & \text{else} \end{cases}$$

and associated arrival vector

$$\underline{A}(n) \equiv (A_{1,1}(n), \cdots, A_{1,N}(n), \cdots A_{M,N}(n))^T.$$

3) The service matrix, indicating which queues are served at slot n

$$\mathbf{S}(n) \equiv [S_{i,j}(n)]$$

where

$$S_{i,j}(n) = \begin{cases} 1, & \text{if } Q_{i,j} \text{ is served at time } n \\ 0, & \text{else} \end{cases}$$

and $\mathbf{S}(n) \in \mathbf{S}$, the set of service[5] matrices.

Note that: $\sum_{i=1}^{M} S_{i,j}(n) = \sum_{j=1}^{N} S_{i,j}(n) \le 1$, and hence if $M = N$, $\mathbf{S}(n) \in \mathbf{S}$ is a *permutation matrix*. If $M \ne N$, we say that $\mathbf{S}(n) \in \mathbf{S}$ is a *quasi-permutation matrix*. We define the associated service vector:

$$\underline{S}(n) \equiv (S_{1,1}(n), \cdots, S_{1,N}(n), \cdots, S_{M,N}(n))^T$$

hence $\|\underline{S}(n)\|^2 \ge \sqrt{NM}$.

4) The *approximate* next-state vector:

$$\tilde{\underline{L}}(n+1) \equiv \underline{L}(n) - \underline{S}(n) + \underline{A}(n)$$

which approximates the exact next-state of each queue

$$L_{i,j}(n+1) = [L_{i,j}(n) - S_{i,j}(n)]^+ + A_{i,j}(n). \quad (4)$$

[5]Note that our definition of the "service" matrix is a permutation matrix which includes the case where an empty queue is served. We adopt this definition here for ease of exposition—it does not affect the result.

B. Proof of Theorem

Before proving the theorem, we first state the following fact and prove the subsequent lemmas.

Fact 1—Birkhoff's Theorem: The doubly substochastic $N \times N$ square matrices form a convex set C with the set of extreme points equal to permutation matrices $\mathbf{S}$.

This is proved in [3].

Lemma 1: The doubly substochastic $M \times N$ nonsquare matrices form a convex set C with the set of extreme points equal to quasi-permutation matrices $\mathbf{S}$.

Proof: Observe that we can add $N - M$ rows to any nonsquare substochastic matrix and introduce new entries so that the row sums of the new rows equal one and further that the column sums are also each one. We can use Birkhoff's Theorem to write the augmented matrix as a convex combination of $N \times N$ permutation matrices. The first M rows of the permutation matrix are an $M \times N$ matrix which forms a permutation matrix with some M of the N columns. The same argument may be applied to additional columns if $M > N$. □

Lemma 2: $\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n)) \le 0, \forall(\underline{L}(n), \underline{\lambda})$, where $\underline{S}^*(n) = \arg\max_{\underline{S}(n)}(\underline{L}^T(n)\underline{S}(n))$, the service matrix selected by the maximum weight matching algorithm to maximize $\underline{L}^T(n)\underline{S}(n)$.

Proof: Consider the linear programming problem:

$$\max(\underline{L}^T(n)\underline{\lambda}) \quad \text{s.t.} \sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \qquad \lambda_{i,j} \ge 0$$

which has a solution equal to an extreme point of the convex set C. Hence,

$$\max(\underline{L}^T(n)\underline{\lambda}) \le \max(\underline{L}^T(n)\underline{S}(n))$$

and so

$$\underline{L}^T(n)\underline{\lambda} - \max(\underline{L}^T(n)\underline{S}(n)) \le 0.$$

□

Lemma 3:

$$E\left[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) \middle| \underline{L}(n)\right] \le 2\sqrt{NM} \qquad \forall \underline{\lambda}.$$

Proof:

$$\begin{aligned} &\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) \\ &\quad = (\underline{L}(n) - \underline{S}(n) + \underline{A}(n))^T(\underline{L}(n) - \underline{S}(n) + \underline{A}(n)) \\ &\qquad - \underline{L}^T(n)\underline{L}(n) \\ &\quad = 2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n)) + (\underline{S}(n) - \underline{A}(n))^T \\ &\qquad \cdot (\underline{S}(n) - \underline{A}(n)) \\ &\quad = 2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n)) + k \end{aligned}$$

where $0 \le k \le 2\sqrt{NM}$ because $\underline{S}(n) - \underline{A}(n)$ is a real vector, and $\|\underline{S}(n) - \underline{A}(n)\|^2 \le 2\sqrt{NM}$.

Taking the expected value:

$$\begin{aligned} &E\left[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) \middle| \underline{L}(n)\right] \\ &\quad \le E\left[2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n))\right] \\ &\quad = 2\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n)) + 2\sqrt{NM}. \end{aligned}$$

From Lemma 2 we know that $2\underline{L}^T(n)(\underline{\lambda}-\underline{S}^*(n)) \le 0$, proving the lemma. □

Lemma 4: $\forall \underline{\lambda} \le (1-\beta)\underline{\lambda}_m$, $0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $||\underline{\lambda}_m||^2 = \min(N, M)$, there exists such that

$$E\left[\underline{\tilde{L}}^T(n+1)\underline{\tilde{L}}(n+1) - \underline{L}^T(n)|\underline{L}(n)\right] \le -\varepsilon||\underline{L}(n)|| + 2\sqrt{NM}.$$

Proof:

$$\begin{aligned}&\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n))\\ &\quad\le \underline{L}^T(n)\{\underline{\lambda}_m - \underline{S}^*(n)\} - \underline{L}^T(n)(\beta\underline{\lambda}_m)\\ &\quad\le 0 - \beta||\underline{L}(n)|| \cdot ||\underline{\lambda}_m|| \cos\theta\end{aligned}$$

where θ is the angle between $\underline{L}(n)$ and $\underline{\lambda}_m$.

We now show that $\cos\theta > \delta$ for some $\delta > 0$ whenever $\underline{L}(n) \ne 0$. First, we show that $\cos\theta > 0$. We do this by contradiction: suppose that $\cos\theta = 0$, i.e., $\underline{L}(n)$ and $\underline{\lambda}_m$ are orthogonal. This can only occur if $\underline{L}(n) = 0$, or if for some i, j, both $\lambda_{i,j} = 0$ and $L_{i,j}(n) > 0$, which is not possible: for arrivals to have occurred at queue $Q_{i,j}$, $\lambda_{i,j}$, must be greater than zero. Therefore, $\cos\theta > 0$ unless $\underline{L}(n) = 0$. Now we show that $\cos\theta$ is bounded away from zero, i.e., that $\cos\theta > \delta$ for some $\delta > 0$. Because $\lambda_{i,j} > 0$ wherever $L_{i,j}(n) > 0$, and because $||\underline{\lambda}||^2 < \sqrt{NM}$

$$\cos\theta = \frac{\underline{L}^T(n)\underline{\lambda}}{||\underline{L}(n)||\,||\underline{\lambda}||} \ge \frac{L_{\max}(n)\lambda_{\min}}{||\underline{L}(n)||(NM)^{1/4}} \tag{5}$$

where $\lambda_{\min} = \min(\lambda_{i,j}, 1 \le i \le M, 1 \le j \le N)$, and $L_{\max}(n) = \max(L_{i,j}(n), 1 \le i \le M, 1 \le j \le N)$.

Also, $||\underline{L}(n)|| \le [NML^2_{\max}(n)]^{1/2} = \sqrt{NM}L_{\max}(n)$, and so $\cos\theta$ is bounded by

$$\cos\theta \ge \frac{\lambda_{\min}}{(NM)^{3/4}}. \tag{6}$$

Therefore

$$E\left[\underline{\tilde{L}}^T(n+1)\underline{\tilde{L}}(n+1) - \underline{L}^T(n)\underline{L}(n)|\underline{L}(n)\right] \le -\frac{\beta\lambda_{\min}}{\sqrt{NM}} + 2\sqrt{NM}.$$

□

Lemma 5: $\forall \underline{\lambda} \le (1-\beta)\underline{\lambda}_m$, $0 < \beta < 1$, there exists $0 < \varepsilon < 1$ such that

$$E\left[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n)|\underline{L}(n)\right] \le -\varepsilon||\underline{L}(n)|| + NM + 2\sqrt{NM}.$$

Proof:

$$L_{i,j}(n+1) = \tilde{L}_{i,j}(n+1) + \begin{cases}1, & \text{if } L_{i,j}(n) = 0,\ S_{i,j}(n) = 1\\ 0, & \text{else}\end{cases}$$

therefore

$$\underline{L}^T(n+1)\underline{L}(n+1) - \underline{\tilde{L}}^T(n+1)\underline{\tilde{L}}(n+1) \le NM \tag{7}$$

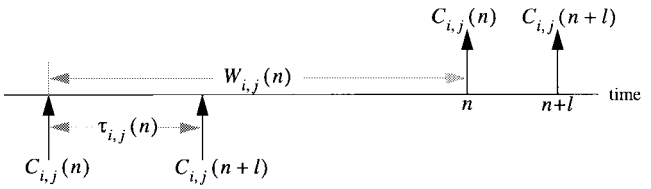


Fig. 5. Arrivals and departures timeline for the VOQ $Q_{i,j}$. Arrivals are shown below the line, departures are shown above the line. $C_{i,j}(n)$ is the current HOL cell at $Q_{i,j}$ which may or may not depart in the current slot, and $C_{i,j}(n+l)$ is the cell that will replace $C_{i,j}(n)$ as HOL cell after it departs.

and so

$$E\left[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n)|\underline{L}(n)\right] \le E\left[\underline{\tilde{L}}^T(n+1)\underline{\tilde{L}}(n+1) - \underline{L}^T(n)\underline{L}(n)|\underline{L}(n)\right] + NM.$$

Using Lemma 4, this concludes the proof. □

Lemma 6: There exists a $V(\underline{L}(n))$ s.t. $E[V(\underline{L}(n+1)) - V(\underline{L}(n))|\underline{L}(n)] \le -\varepsilon||\underline{L}(n)|| + k$, where $k, \varepsilon > 0$.

Proof: $V(\underline{L}(n)) = \underline{L}^T(n)\underline{L}(n)$ and $k = NM + 2\sqrt{NM}$ in Lemma 5. □

We are now ready to prove the main theorem. $V(\underline{L}(n))$ in the main theorem is a quadratic Lyapunov function and, according to the argument of Kumar and Meyn [13], it follows that the switch is stable.

APPENDIX B
OCF PROOF

A. Definitions

In addition to the definitions defined in Appendix A, the following definitions are necessary in this Appendix. Consider Fig. 5.

1) $C_{i,j}(n)$ denotes the HOL cell of $Q_{i,j}$ at slot n.
2) The interarrival time vector:

$$\underline{\tau}(n) \equiv (\tau_{1,1}(n), \cdots, \tau_{1,N}(n), \cdots, \tau_{M,1}(n), \cdots, \tau_{M,N}(n)) \tag{8}$$

where $\tau_{i,j}(n)$ is the interarrival time between $C_{i,j}(n)$ and the cell behind it in line.
3) The waiting time vector:

$$\underline{W}(n) \equiv (W_{1,1}(n), \cdots, W_{1,N}(n), \cdots, W_{M,1}(n), \cdots, W_{M,N}(n)) \tag{9}$$

where $W_{i,j}(n)$ is the waiting time of $C_{i,j}(n)$ at slot n.
4) The positive-definite diagonal matrix T whose diagonal elements are $\{\lambda_{1,1}, \cdots, \lambda_{1,N}, \cdots, \lambda_{M,1}, \cdots, \lambda_{M,N}\}$.
5) $[\underline{a} \cdot \underline{b} \cdot \underline{c}]$ denotes a vector in which each element is a product of the corresponding elements of the vectors: $\underline{a}$, $\underline{b}$, and $\underline{c}$, i.e., $a_{i,j} \cdot b_{i,j} \cdot c_{i,j}$.
6) The approximate waiting time next-state vector:

$$\underline{\tilde{W}}(n+1) \equiv \underline{W}(n) + \underline{1} - [\underline{S}(n) \cdot \underline{\tau}(n)]. \tag{10}$$

B. Proof of Theorem

The proof consists of two steps. First, we prove the stability of the waiting time. Then, we show that the stability of the waiting time implies the stability of queue occupancy. But before proving the theorem, we first state the following facts and prove the subsequent lemmas.

Fact 2: An interarrival time $\tau_{i,j}(n)$ is independent of a waiting time $W_{i,j}(n) \forall i, j, n$.

Fact 3: $\tau_{i,j}(n) \geq 1$. Since there is only at most one arrival per slot, the arrival time of any two consecutive cells must be at least one slot apart.

Fact 4: $W_{i,j}(n) \geq L_{i,j}(n) \forall i, j, n$ because there is at most one arrival per slot.

Fact 5: For any queue $Q_{i,j}$ whose arrival rate is zero, $\lambda_{i,j} = 0$, $L_{i,j}(n) = 0$, thus $W_{i,j}(n) = 0 \forall n$. Considering the fact that a zero waiting time does not contribute to the sum value $\underline{W}^T(n)\underline{S}(n)$, without loss of generality, we can set the corresponding service indicator $S_{i,j}(n)$ to zero for all time $S_{i,j}(n) = 0 \, forall \, n$.

Lemma 7: $\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \leq 0 \forall \underline{W}(n)$, λ, where $\underline{S}^*(n)$ is such that $\underline{W}^T(n)\underline{S}^*(n) = \max(\underline{W}^T(n)\underline{S}(n))$.

Proof: The proof is similar to the proof of Lemma 2 in Appendix A. □

Lemma 8: $\forall \underline{\lambda} \leq (1-\beta)\underline{\lambda}_m$, $0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $\|\underline{\lambda}_m\|^2 = \min(N, M) \leq \sqrt{NM}$, there exists $0 < \varepsilon < 1$ such that

$$E\Big[\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\Big] \leq -\varepsilon\|\underline{W}(n)\| + K.$$

Proof: By expansion

$$\begin{aligned}
&\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) \\
&\quad = (\underline{W}(n) + \underline{1} - [\underline{S}^*(n)\cdot\underline{\tau}(n)])^T T(\underline{W}(n) + \underline{1} \\
&\qquad - [\underline{S}^*(n)\cdot\underline{\tau}(n)]) \\
&\quad = \underline{W}^T(n)T\underline{W}(n) + 2\underline{W}^T(n)\underline{\lambda} - 2\underline{W}^T(n) \\
&\qquad \cdot [\underline{S}^*(n)\cdot\underline{\tau}(n)\cdot\underline{\lambda}] \\
&\qquad + \sum_{i,j}\lambda_{i,j} - 2\sum_{i,j} S^*_{i,j}(n)\cdot\tau_{i,j}(n)\cdot\lambda_{i,j} \\
&\qquad + \sum_{i,j} S^*_{i,j}(n)\cdot\tau^2_{i,j}(n)\cdot\lambda_{i,j}. \qquad (11)
\end{aligned}$$

Subtracting $\underline{W}^T(n)T\underline{W}(n)$ from both sides and taking the expected value

$$\begin{aligned}
&E\Big[\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\Big] \\
&\quad = 2\big(\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n)\big) + \sum_{i,j}\lambda_{i,j} \\
&\qquad - 2\sum_{i,j} S^*_{i,j}(n) + \sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}}. \qquad (12)
\end{aligned}$$

After imposing the admissibility constraints and the scheduling algorithm properties, we obtain the following inequalities:

$$\sum_{i,j}\lambda_{i,j} < N, \quad \sum_{i,j} S^*_{i,j}(n) \geq 0, \quad \sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}} \leq L < \infty \qquad (13)$$

where L is a nonnegative constant.

From (12) and (13), we obtain

$$E\Big[\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\Big] \leq 2\big(\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n)\big) + L + N. \qquad (14)$$

From the relationship of the arrival vector

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \leq \underline{W}^T(n)(1-\beta)\underline{\lambda}_m - \underline{W}^T(n)\underline{S}^*(n). \qquad (15)$$

Applying Lemma 7

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \leq -\beta\underline{W}^T(n)\underline{\lambda}_m, \qquad (16)$$

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \leq -\beta\|\underline{W}(n)\|\cdot\|\underline{\lambda}_m\|\cdot\cos\theta \qquad (17)$$

where θ is the angle between $\underline{W}(n)$ and $\underline{\lambda}_m$.

Using the same approach as in Lemma 4, it follows that

$$\cos\theta \geq \frac{\lambda_{\min}}{(NM)^{3/4}}. \qquad (18)$$

Using (14), (17), and (18)

$$E\Big[\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\Big] \leq -\beta\frac{\lambda_{\min}}{\sqrt{NM}}\|\underline{W}(n)\| + K \qquad (19)$$

where $\varepsilon = \beta(\lambda_{\min}/\sqrt{NM})$. □

Lemma 9: $\forall \underline{\lambda} \leq (1-\beta)\underline{\lambda}_m$, $0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $\|\underline{\lambda}_m\|^2 = \min(N, M) \leq \sqrt{NM}$, there exists $0 < \varepsilon < 1$ such that

$$E\big[\underline{W}^T(n+1)T\underline{W}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\big] \leq -\varepsilon\|\underline{W}(n)\| + K.$$

Proof: We can draw the following relationship between the two waiting times:

$$W_{i,j}(n+1) = \begin{cases} \tilde{W}_{i,j}(n+1), & \tilde{W}_{i,j}(n+1) \geq 0 \\ 0, & \tilde{W}_{i,j}(n+1) < 0. \end{cases} \qquad (20)$$

Since T is a positive definite matrix, (20) implies

$$\underline{W}^T(n+1)T\underline{W}(n+1) \leq \tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) \, forall \, n. \qquad (21)$$

Hence

$$E\big[\underline{W}^T(n+1)T\underline{W}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\big] \leq E\Big[\tilde{\underline{W}}^T(n+1)T\tilde{\underline{W}}(n+1) - \underline{W}^T(n)T\underline{W}(n)|\underline{W}(n)\Big]. \qquad (22)$$

This proves the Lemma. □

Lemma 10: There exists a quadratic Lyapunov function $V(\underline{W}(n))$ such that

$$E[V(\underline{W}(n+1)) - V(\underline{W}(n))|\underline{W}(n))] \leq -\varepsilon\|\underline{W}(n)\| + K \qquad (23)$$

where K is a constant and $\varepsilon > 0$.

Proof: From Lemma 9 $V(\underline{W}(n)) = \underline{W}^T(n)T\underline{W}(n), \varepsilon = \beta(\lambda_{\min}/\sqrt{NM}) > 0$, and $K = L + N > 0$. □

Theorem 3: Under the OCF algorithm, the waiting times are stable for all admissible and independent arrival processes, i.e., $E([\|\underline{W}(n)\|] < \infty)$.

Proof: Similar to the argument in the LQF proof. □

Theorem 4: Under the OCF algorithm, the queue occupancies are stable for all admissible and independent arrival processes, i.e., $E([||\underline{L}(n)||] < \infty)$.

Proof: From Fact 4, $W_{i,j}(n) \geq L_{i,j}(n) \, for all \, i, j, n.$ Thus,

$$E[||\underline{L}(n)||] \leq E[||\underline{W}(n)||] < \infty. \tag{24}$$

□

REFERENCES

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

[2] M. Ali and H. Nguyen, "A neural network implementation of an input access scheme in a high-speed packet switch," in *Proc. GLOBECOM 1989,* pp. 1192–1196.

[3] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser.,* vol. A5, pp. 147–150, 1946.

[4] T. X. Brown and K. H. Liu, "Neural network design of a Banyan network controller," *IEEE J. Select. Areas Commun.,* vol. 8, pp. 1289–1298, Oct. 1990.

[5] M. Chen and N. D. Georganas, "A fast algorithm for multi-channel/port traffic scheduling," in *Proc. IEEE Supercom/ICC'94,* pp. 96–100.

[6] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math. Dokl.,* vol. 11, pp. 1277–1280, 1970.

[7] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," *Soc. Ind. Appl. Math. J. Comput.,* vol. 2, pp. 225–231, 1973.

[8] A. Huang and S. Knauer, "Starlite: A wideband digital switch," in *Proc. GLOBECOM'84,* pp. 121–125.

[9] M. Karol, K. Eng, and H. Obara, "Improving the performance of input-queued ATM packet switches," in *INFOCOM'92,* pp. 110–115.

[10] M. Karol and M. Hluchyj, "Queuing in high-performance packet-switching," *IEEE J. Select. Areas Commun.,* vol. 6, pp. 1587–1597, Dec. 1988.

[11] M. J. Karol, M. Hluchyj, and S. Morgan, "Input vs. output queuing on a space-division packet switch," in *Proc. GLOBECOM 1986,* pp. 659–665.

[12] ———, "Input versus output queuing on a space division switch," *IEEE Trans. Commun.,* vol. COM-35, pp. 1347–1356, Dec. 1987.

[13] P. R. Kumar and S. P. Meyn, "Stability of queuing networks and scheduling policies," *IEEE Trans. Automat. Contr.,* vol. 40, Feb. 1995.

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

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

[16] H. Obara, "Optimum architecture for input queuing ATM switches," *IEE Electron. Lett.,* pp. 555–557, Mar. 28, 1991.

[17] H. Obara and Y. Hamazumi, "Parallel contention resolution control for input queuing ATM switches," *Electron. Lett.,* vol. 28, no. 9, pp. 838–839, Apr. 23, 1992.

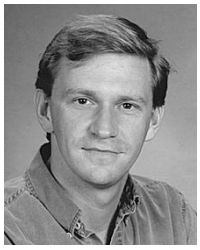
[18] H. Obara, S. Okamoto, and Y. Hamazumi, "Input and output queuing ATM switch architecture with spatial and temporal slot reservation control," *Electron. Lett.,* pp. 22–24, Jan. 2, 1992.

[19] Y. Tamir and G. Frazier, "High performance multi-queue buffers for VLSI communication switches," in *Proc. 15th Ann. Symp. Computer Architecture,* June 1988, pp. 343–354.

[20] R. E. Tarjan, "Data structures and network algorithms," *Soc. Ind. Appl. Math.,* Philadelphia, PA, Nov. 1983.

[21] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.,* vol. 37, pp. 1936–1948, Dec. 1992.

[22] T. P. Troudet and S. M. Walters, "Hopfield neural network architecture for crossbar switch control," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 42–57, Jan. 1991.



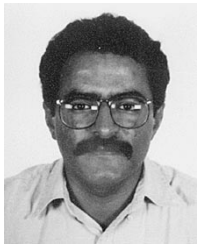
Nick McKeown (S'91–M'95–SM'97) completed the Ph.D. degree at the University of California at Berkeley in 1995.

He is a Professor of electrical engineering and computer science at Stanford University, Stanford, CA, where he works on the theory, design, and implementation of high-speed Internet routers and switches. He has worked for Hewlett-Packard Labs, Cisco Systems, and has an active consulting business.

Dr. McKeown is as an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS, the Robert Noyce Faculty Fellow at Stanford, and is a Research Fellow of the Alfred P. Sloan Foundation.

Adisak Mekkittikul (S'87–M'98) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA.

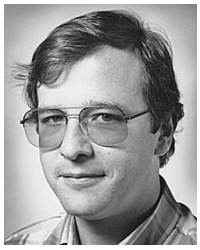
He is a Senior Member of Technical Staff at Berkeley Concept Research Corp. His research interests include high-speed switches and routers and wireless networks. Previously, he worked for Hewlett-Packard, Cirrus Logic, and Trident Microsystems.



Venkat Anantharam (M'86–SM'96–F'98) received the B.Tech. degree in electronics in 1980 from the Indian Institute of Technology at Madras (now Chennai) and the M.A. and C.Phil. degrees in mathematics and the M.S. and Ph.D. degrees in electrical engineering in 1983, 1984, 1982, and 1986, respectively, from the University of California at Berkeley.

From 1986 to 1994, he was on the faculty of the School of Electrical Engineering at Cornell University, Ithaca, NY. Since 1994, he has been on the faculty of the Electrical Engineering and Computer Science Department at the University of California at Berkeley.

Dr. Anantharam is a recipient of the Philips India Medal (1980), the President of India Gold Medal (1980), the NSF Presidential Young Investigator Award (1988), the IBM Faculty Development Award (1989), and co-recipient (with S. Verdú) of the Information Theory Society Paper Award (1998). He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY, queueing systems: theory and applications, and Markov processes and related fields.



Jean Walrand (S'71–M'74–SM'90–F'93) received the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley.

He is a Professor in the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley. His research interests include decision theory, stochastic processes, and communication networks. He is the author of *An Introduction to Queuing Networks* (Prentice Hall, 1988), *Communication Networks: A First Course* (2nd ed., McGraw-Hill, 1998), and co-author of *High-Performance Communication Networks* (Morgan Kaufman, 1996).

Prof. Walrand is a Fellow of the Belgian American Education Foundation and a recipient of the Lanchester Prize.