Input Versus Output Queueing on a Space-Division Packet Switch

MARK J. KAROL, MEMBER, IEEE, MICHAEL G. HLUCHYJ, MEMBER, IEEE, AND SAMUEL P. MORGAN, FELLOW, IEEE

Abstract—**Two simple models of queueing on an $N \times N$ space-division packet switch are examined. The switch operates synchronously with fixed-length packets; during each time slot, packets may arrive on any inputs addressed to any outputs. Because packet arrivals to the switch are unscheduled, more than one packet may arrive for the same output during the same time slot, making queueing unavoidable. Mean queue lengths are always greater for queueing on inputs than for queueing on outputs, and the output queues saturate only as the utilization approaches unity. Input queues, on the other hand, saturate at a utilization that depends on N, but is approximately $(2 - \sqrt{2}) = 0.586$ when N is large. If output trunk utilization is the primary consideration, it is possible to slightly increase utilization of the output trunks—up to $(1 - e^{-1}) = 0.632$ as $N \to \infty$—by dropping interfering packets at the end of each time slot, rather than storing them in the input queues. This improvement is possible, however, only when the utilization of the input trunks exceeds a second critical threshold—approximately $\ln (1 + \sqrt{2}) = 0.881$ for large N.**

I. INTRODUCTION

SPACE-DIVISION packet switching is emerging as a key component in the trend toward high-performance integrated communication networks for data, voice, image, and video [1], [2] and multiprocessor interconnects for building highly parallel computer systems [3], [4]. Unlike present-day packet switch architectures with throughputs measured in 1's or at most 10's of Mbits/s, a space-division packet switch can have throughputs measured in 1's, 10's, or even 100's of Gbits/s. These capacities are attained through the use of a highly parallel switch fabric coupled with simple per packet processing distributed among many high-speed VLSI circuits.

Conceptually, a space-division packet switch is a box with N inputs and N outputs that routes the packets arriving on its inputs to the appropriate outputs. At any given time, internal switch points can be set to establish certain paths from inputs to outputs; the routing information used to establish input–output paths is often contained in the header of each arriving packet. Packets may have to be buffered within the switch until appropriate connections are available; the location of the buffers and the amount of buffering required depend on the switch architecture and the statistics of the offered traffic.

Clearly, congestion can occur if the switch is a blocking network, that is, if there are not enough switch points to provide simultaneous, independent paths between arbitrary pairs of inputs and outputs. A Banyan switch [3]–[5], for example, is a blocking network. In a Banyan switch, even when every input is assigned to a different output, as many as $\sqrt{N}$ connections may be contending for use of the same center link. The use of a blocking network as a packet switch is feasible only under light loads or, alternatively, if it is possible to run the switch substantially faster than the input and output trunks.

In this paper, we consider only nonblocking networks. A simple example of a nonblocking switch fabric is the crossbar interconnect with N^2 switch points (Fig. 1). Here it is always possible to establish a connection between any idle input–output pair. Examples of other nonblocking switch fabrics are given in [3]. Even with a nonblocking interconnect, some queueing in a packet switch is unavoidable, simply because the switch acts as a statistical multiplexor; that is, packet arrivals to the switch are unscheduled. If more than one packet arrives for the same output at a given time, queueing is required. Depending on the speed of the switch fabric and its particular architecture, there may be a choice as to where the queueing is done: for example, on the input trunk, on the output trunk, or at an internal node.

We assume that the switch operates synchronously with fixed-length packets, and that during each time slot, packets may arrive on any inputs addressed to any outputs (Fig. 2). If the switch fabric runs N times as fast as the input and output trunks, all the packets that arrive during a particular input time slot can traverse the switch before the next input slot, but there will still be queueing at the outputs [Fig. 1(a)]. This queueing really has nothing to do with the switch architecture, but is due to the simultaneous arrival of more than one input packet for the same output. If, on the other hand, the switch fabric runs at the same speed as the inputs and outputs, only one packet can be accepted by any given output line during a time slot, and other packets addressed to the same output must queue on the input lines [Fig. 1(b)]. For simplicity, we do not consider the intermediate case where some packets can be queued at internal nodes, as in the Banyan topology.

It seems intuitively reasonable that the mean queue lengths, and hence the mean waiting times, will be greater for queueing on inputs than for queueing on outputs. When queueing is done on inputs, a packet that could traverse the switch to an idle output during the current time slot may have to wait in queue behind a packet whose output is currently busy. The intuition that, if possible, it is better to queue on the outputs than the inputs of a space-division packet switch also pertains to the following situation. Consider a single road leading to both a sports arena and a store [Fig. 3(a)]. Even if there are no customers waiting for service in the store, some shoppers might be stuck in stadium traffic. A simple bypass road around the stadium is the obvious solution [Fig. 3(b)].

This paper quantifies the performance improvements provided by output queueing for the following simple model. Independent, statistically identical traffic arrives on each input trunk. In any given time slot, the probability that a packet will arrive on a particular input is p. Thus, p represents the average utilization of each input. Each packet has equal probability $1/N$ of being addressed to any given output, and successive packets are independent.

With output queueing, all arriving packets in a time slot are

Paper approved by the Editor for Local Area Networks of the IEEE Communications Society. Manuscript received August 8, 1986; revised May 14, 1987. This paper was presented at GLOBECOM'86, Houston, TX, December 1986.

M. J. Karol is with AT&T Bell Laboratories, Holmdel, NJ 07733.

M. G. Hluchyj was with AT&T Bell Laboratories, Holmdel, NJ 07733. He is now with the Codex Corporation, Canton, MA 02021.

S. P. Morgan is with AT&T Bell Laboratories, Murray Hill, NJ 07974.

IEEE Log Number 8717486.

0090-6778/87/1200-1347$01.00

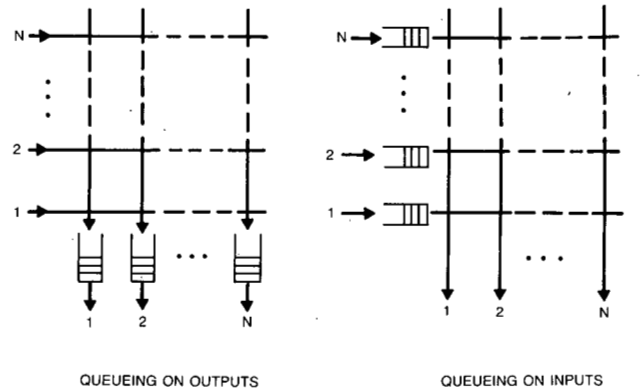

Fig. 1. (a) An $N \times N$ crossbar switch with output queueing. (b) An $N \times N$ crossbar switch with input queueing.

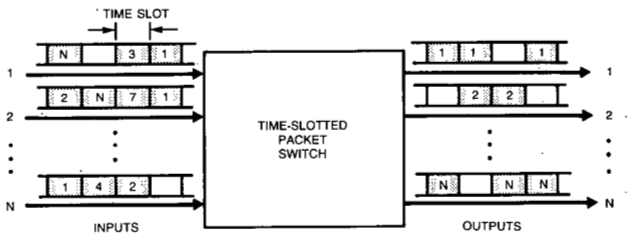

Fig. 2. Fixed-length packets arrive synchronously to a time-slotted packet switch.

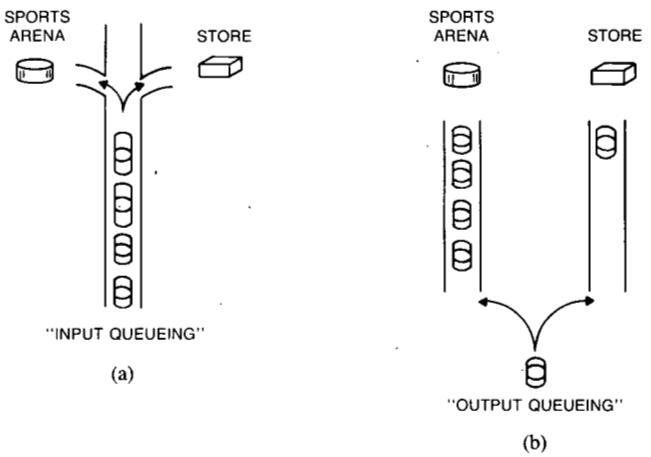

Fig. 3. "Output queueing" (b) is superior to "input queueing" (a). In (a), even if there are no customers waiting for service in the store, some shoppers might be stuck in stadium traffic. In (b), a bypass road around the stadium serves cars traveling to the store.

cleared before the beginning of the next time slot. For example, a crossbar switch fabric that runs N times as fast as the inputs and outputs can queue all packet arrivals according to their output addresses, even if all N inputs have packets destined for the same output [Fig. 1(a)]. If k packets arrive for one output during the current time slot, however, only one can be transmitted over the output trunk. The remaining $k - 1$ packets go into an output FIFO (first-in, first-out queue) for transmission during subsequent time slots. Since the average utilization of each output trunk is the same as the utilization of each input trunk, namely p, the system is stable and the mean queue lengths will be finite for $p < 1$, but they will be greater than zero if $p > 0$.

A crossbar interconnect with the switch fabric running at the same speed as the inputs and outputs exemplifies input queueing [Fig. 1(b)]. Each arriving packet goes, at least momentarily, into a FIFO on its input trunk. At the beginning of every time slot, the switch controller looks at the first packet in each FIFO. If every packet is addressed to a different output, the controller closes the proper crosspoints and all the packets go through. If k packets are addressed to a particular output, the controller picks one to send; the others wait until the next time slot, when a new selection is made among the packets that are then waiting. Three selection policies are discussed in Section III: one of the k packets is chosen at *random*, each selected with equal probability $1/k$, *longest queue* selection, in which the controller sends the packet from the longest input queue,[1] and *fixed priority* selection where the N inputs have fixed priority levels, and of the k packets, the controller sends the one with highest priority.

Solutions of these two queueing problems are given in Sections II and III. Curves showing mean waiting time as a function of p are plotted for various values of N. As expected, the mean waiting times are greater for queueing on inputs than for queueing on outputs. Furthermore, the output queues saturate only as $p \to 1$. Input queues, on the other hand, saturate at a value of p less than unity, depending weakly on N; for large N, the critical value of p is approximately $(2 - \sqrt{2}) = 0.586$ with the random selection policy. When the utilization p of the input trunks exceeds the critical value, the steady-state queue sizes are infinite, packets experience infinite waiting times, and the output trunk utilization is limited to approximately 0.586 (for large N). In the saturation region, however, it is possible to increase utilization of the output trunks—up to $(1 - e^{-1}) = 0.632$ as $N \to \infty$—by dropping packets, rather than storing them in the input queues. This improvement is possible, however, only when the utilization of the input trunks exceeds a second critical threshold—approximately $\ln (1 + \sqrt{2}) = 0.881$ for large N. Consequently, if the objective is maximum output utilization, rather than 100 percent packet delivery, then below the second threshold, it is better to queue packets until they are successful, whereas above the second threshold, it is better to reduce input queue blocking by dropping packets whenever there are conflicts. With high probability, new packets (with new destinations) will quickly arrive to replace the dropped packets.

Comparing the random and longest queue selection policies of input queueing, the mean waiting times are greater with random selection. This is expected because the longest queue selection policy reduces the expected number of packets blocked (behind other packets) from traversing the switch to idle outputs. For fairness, the fixed priority discipline should be avoided because the lowest priority input queue suffers large delays and is sometimes unstable, even when the other two selection policies guarantee stability.

II. Queues on Outputs

Much of the following analysis of the output queueing scheme involves well-known results for discrete-time queueing systems [6]. Communication systems have been modeled by discrete-time queues in the past (e.g., [7]); we sketch our analysis and present results for later comparison to the input queueing analysis.

We assume that packet arrivals on the N input trunks are governed by independent and identical Bernoulli processes. Specifically, in any given time slot, the probability that a packet will arrive on a particular input is p. Each packet has equal probability $1/N$ of being addressed to any given output, and successive packets are independent.

Fixing our attention on a particular output queue (the

[1] A random selection is made if, of the k input queues with packets addressed to a particular output, several queues have the same maximum length.

"tagged" queue), we define the random variable A as the number of packet arrivals at the tagged queue during a given time slot.[2] It follows that A has the binomial probabilities

$$a_i \triangleq \Pr[A=i]=\binom{N}{i}(p/N)^i(1-p/N)^{N-i} \qquad i=0, 1, \cdots, N \quad (1)$$

with probability generating function (PGF)

$$A(z) \triangleq \sum_{i=0}^{N} z^i \Pr[A=i]=\left(1-\frac{p}{N}+z\frac{p}{N}\right)^N. \quad (2)$$

As $N \to \infty$, the number of packet arrivals at the tagged queue during each time slot has the Poisson probabilities

$$a_i \triangleq \Pr[A=i]=\frac{p^i e^{-p}}{i!} \qquad i=0, 1, 2, \cdots \quad (3)$$

with probability generating function (PGF)

$$A(z) \triangleq \sum_{i=0}^{N} z^i \Pr[A=i]=e^{-p(1-z)}. \quad (4)$$

Letting Q_m denote the number of packets in the tagged queue at the end of the mth time slot, and A_m denote the number of packet arrivals during the mth time slot, we have

$$Q_m=\max(0, Q_{m-1}+A_m-1). \quad (5)$$

When $Q_{m-1} = 0$ and $A_m > 0$, one of the new packets is immediately transmitted during the mth time slot; that is, a packet flows through the switch without suffering any delay. The queue size Q_m is modeled by a discrete-time Markov chain; Fig. 4 illustrates the state transition diagram. Using (5) and following a standard approach in queueing analysis (see, for example, [8, sect. 5.6]), we obtain the PGF for the steady-state queue size:

$$Q(z)=\frac{(1-p)(1-z)}{A(z)-z}. \quad (6)$$

Finally, substituting the right-hand side of (2) into (6), we obtain

$$Q(z)=\frac{(1-p)(1-z)}{\left(1-\frac{p}{N}+z\frac{p}{N}\right)^N-z}. \quad (7)$$

Now, differentiating (7) with respect to z and taking the limit as $z \to 1$, we obtain the mean steady-state queue size $\bar{Q}$ given by

$$\bar{Q}=\frac{(N-1)}{N}\cdot\frac{p^2}{2(1-p)}=\frac{(N-1)}{N}\bar{Q}_{M/D/1} \quad (8)$$

where $\bar{Q}_{M/D/1}$ denotes the mean queue size for an $M/D/1$ queue. Hence, as $N \to \infty$, $\bar{Q} \to \bar{Q}_{M/D/1}$.

We can make the even stronger statement that the steady-state probabilities for the queue size converge to those of an $M/D/1$ queue. Taking the limit as $N \to \infty$ on both sides of (7) yields

$$\lim_{N\to\infty} Q(z)=\frac{(1-p)(1-z)}{e^{-p(1-z)}-z} \quad (9)$$

[2] We use the phrase "arrivals at the tagged queue *during* a given time slot" to indicate that packets do not arrive instantaneously, in their entirety, at the output. Packets have a nonzero transmission time.

STATE TRANSITION PROBABILITIES

$a_i \equiv \Pr(A=i) \qquad i = 0, 1, 2, \ldots$

$$= \begin{cases} \frac{p^i e^{-p}}{i!} & \text{IF } N = \infty \\ \binom{N}{i}\left(\frac{p}{N}\right)^i\left(1-\frac{p}{N}\right)^{N-i} & \text{IF } N < \infty \end{cases}$$

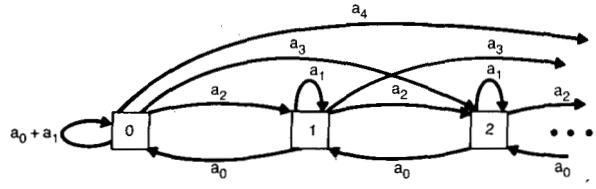

Fig. 4. The discrete-time Markov chain state transition diagram for the output queue size.

which corresponds to the PGF for the steady-state queue size of an $M/D/1$ queue. Expanding (9) in a Maclaurin series [9] yields the asymptotic (as $N \to \infty$) queue size probabilities[3]

$$\Pr(Q=0)=(1-p)e^p \quad (10)$$

$$\Pr(Q=1)=(1-p)e^p(e^p-1-p) \quad (11)$$

$$\vdots$$

$$\Pr(Q=n)=(1-p)\sum_{j=1}^{n+1}(-1)^{n+1-j}e^{jp} \cdot \left[\frac{(jp)^{n+1-j}}{(n+1-j)!}+\frac{(jp)^{n-j}}{(n-j)!}\right] \qquad \text{for } n \geq 2 \quad (12)$$

where the second factor in (12) is ignored for $j = (n + 1)$.

Although it is mathematically pleasing to have closed-form expressions, directly using (12) to compute the steady-state probabilities leads to inaccurate results for large n. When n is large, the alternating series (12) expresses small steady-state probabilities as the difference between very large positive numbers. Accurate values are required if one is interested in the tail of the distribution; for example, to compute the probability that the queue size exceeds some value M. Numerically, a more accurate algorithm is obtained directly from the Markov chain (Fig. 4) balance equations. Equations (13)–(15) numerically provide the steady-state queue size probabilities.

$$q_0 \triangleq \Pr(Q=0)=\frac{(1-p)}{a_0} \quad (13)$$

$$q_1 \triangleq \Pr(Q=1)=\frac{(1-a_0-a_1)}{a_0}\cdot q_0 \quad (14)$$

$$\vdots$$

$$q_n \triangleq \Pr(Q=n)=\frac{(1-a_1)}{a_0}\cdot q_{n-1} - \sum_{i=2}^{n}\frac{a_i}{a_0}\cdot q_{n-i} \qquad n \geq 2 \quad (15)$$

[3] The steady-state probabilities in [9, sect. 5.1.5] are for the *total number* of packets in an $M/D/1$ system. We are interested in *queue size;* hence, the modification to (10)–(12).

where the a_i are given by (1) and (3) for $N < \infty$ and $N = \infty$, respectively.

We are now interested in the waiting time for an arbitrary (tagged) packet that arrives at the tagged output FIFO during the mth time slot. We assume that packet arrivals to the output queue in the mth time slot are transmitted over the output trunk in random order. All packets arriving in earlier time slots, however, must be transmitted first.

The tagged packet's waiting time W has two components. First, the packet must wait W_1 time slots while packets that arrived in earlier time slots are transmitted. Second, it must wait an additional W_2 time slots until it is randomly selected out of the packet arrivals in the mth time slot.

Since packets require one time slot for transmission over the output trunk, W_1 equals Q_{m-1}. Consequently, from (6), the PGF for the steady-state value of W_1 is

$$W_1(z) = \frac{(1-p)(1-z)}{A(z)-z}. \tag{16}$$

We must be careful when we compute W_2, the delay due to the transmission of other packet arrivals in the mth time slot. Burke points out in [10] that many standard works on queueing theory are in error when they compute the delay of single-server queues with batch input. Instead of working with the size of the batch to which the tagged packet belongs, it is tempting to work with the size of an arbitrary batch. Errors result when the batches are not of constant size. The probability that our tagged packet arrives in a batch of size i is given by $ia_i/\bar{A}$; hence, the random variable W_2 has the probabilities

$$\Pr[W_2 = k] = \sum_{i=k+1}^{\infty} \frac{1}{i}\, ia_i/\bar{A} \qquad k = 0, 1, 2, \cdots$$

$$= \frac{1}{p} \sum_{i=k+1}^{\infty} a_i \tag{17}$$

where $\bar{A}$ $(=p)$ is the expected number of packet arrivals at the tagged output during each time slot, and the a_i are given by (1) and (3) for $N < \infty$ and $N = \infty$, respectively. The PGF for the steady-state value of W_2 follows directly from (17).

$$W_2(z) = \frac{1-A(z)}{p(1-z)}. \tag{18}$$

Finally, since W is the sum of the independent random variables W_1 and W_2, the PGF for the steady-state waiting time is

$$W(z) = Q(z) \cdot \frac{1-A(z)}{p(1-z)}. \tag{19}$$

$A(z)$ is given by (2) and (4) for $N < \infty$ and $N = \infty$, respectively.

Differentiating (19) with respect to z and taking the limit as $z \to 1$, we obtain the mean steady-state waiting time given by

$$\bar{W} = \bar{Q} + \frac{1}{2p}[\bar{A}^2 - \bar{A}]. \tag{20}$$

Since $\bar{A} = p$ and $\bar{A}^2 = p^2 + p(1 - p/N)$, substituting the right-hand side of (8) into (20) yields

$$\bar{W} = \frac{(N-1)}{N} \cdot \frac{p}{2(1-p)} = \frac{(N-1)}{N}\, \bar{W}_{M/D/1} \tag{21}$$

where $\bar{W}_{M/D/1}$ denotes the mean waiting time for an $M/D/1$ queue. The mean waiting time $\bar{W}$, as a function of p, is shown in Fig. 5 for several values of N. Notice that Little's result and (8) generate the same formula for $\bar{W}$.

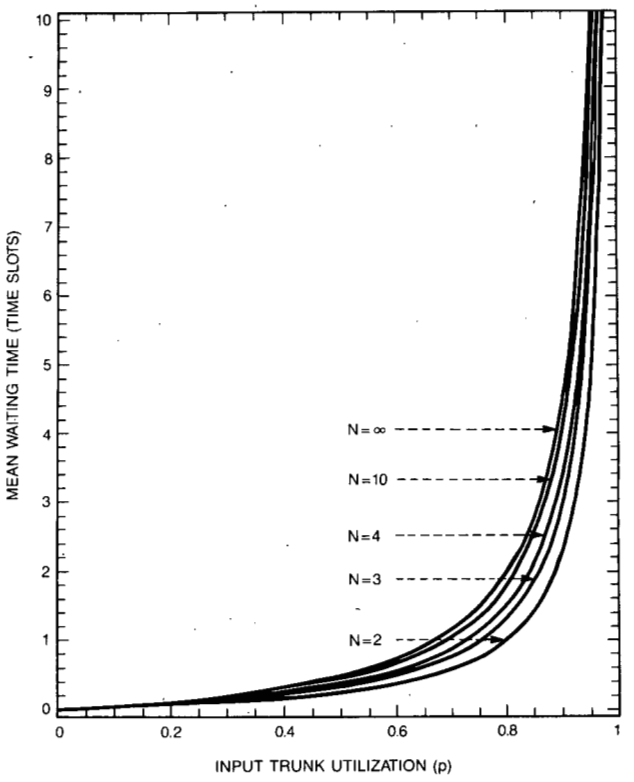

Fig. 5. The mean waiting time for several switch sizes N with output queueing.

Rather than take the inverse transform of $W(z)$, it is easier to compute the steady-state waiting time probabilities from

$$\Pr[W = k] = \Pr[W_1 + W_2 = k]$$

$$= \frac{1}{p} \sum_{n=0}^{k} q_n \cdot \sum_{i=k+1-n}^{\infty} a_i \qquad k = 0, 1, \cdots$$

$$= \frac{1}{p} \sum_{n=0}^{k} q_n \cdot \left[1 - \sum_{i=0}^{k-n} a_i\right] \tag{22}$$

where the q_n are given by (13)–(15) and the a_i are given by (1) and (3) for $N < \infty$ and $N = \infty$, respectively.

III. Queues on Inputs

The interesting analysis occurs when the switch fabric runs at the same speed as the input and output trunks, and packets are queued at the inputs. How much traffic can the switch accommodate before it saturates, and how much does the mean waiting time increase when we queue packets at the inputs rather than at the outputs? As in the previous section, we assume that packet arrivals on the N input trunks are governed by independent and identical Bernoulli processes. In any given time slot, the probability that a packet will arrive on a particular input is p; each packet has equal probability $1/N$ of being addressed to any given output. Each arriving packet goes, at least momentarily, into a FIFO on its input trunk. At the beginning of every time slot, the switch controller looks at the first packet in each FIFO. If every packet is addressed to a different output, the controller closes the proper crosspoints and all the packets go through. If k packets are addressed to a particular output, one of the k packets is chosen at *random*, each selected with equal probability $1/k$. The others wait until the next time slot when a new selection is made among the packets that are then waiting.

A. Saturation Analysis—Random Selection Policy

Suppose the input queues are saturated so that packets are always waiting in every input queue. Whenever a packet is transmitted through the switch, a new packet immediately replaces it at the head of the input queue. We define B_m^i as the number of packets at the heads of input queues that are "blocked" for output i at the end of the mth time slot. In other words, B_m^i is the number of packets destined for output i, but not selected by the controller during the mth time slot. We also define A_m^i as the number of packets moving to the head of "free" input queues during the mth time slot and destined for output i. An input queue is "free" during the mth time slot if, and only if, a packet from it was transmitted during the $(m-1)$st time slot. The new packet "arrival" at the head of the queue has equal probability $1/N$ of being addressed to any given output. It follows that

$$B_m^i = \max\,(0,\, B_{m-1}^i + A_m^i - 1). \tag{23}$$

Although B_m^i does not represent the occupancy of any physical queue, notice that (23) has the same mathematical form as (5).

A_m^i, the number of packet arrivals during the mth time slot to free input queues and destined for output i, has the binomial probabilities

$$\Pr\,[A_m^i = k] = \binom{F_{m-1}}{k} (1/N)^k (1-1/N)^{F_{m-1}-k}$$
$$k = 0,\, 1,\, \cdots,\, F_{m-1} \tag{24}$$

where

$$F_{m-1} \triangleq N - \sum_{i=1}^{N} B_{m-1}^i. \tag{25}$$

F_{m-1} is the number of free input queues at the end of the $(m-1)$st time slot, representing the total number of packets transmitted through the switching during the $(m-1)$st time slot. Therefore, F_{m-1} is also the total number of input queues with new packets at their heads during the mth time slot. That is,

$$F_{m-1} = \sum_{i=1}^{N} A_m^i. \tag{26}$$

Notice that $\bar{F}/N = \rho_0$ where $\bar{F}$ is the mean steady-state number of free input queues and ρ_0 is the utilization of the output trunks (i.e., the switch throughput). As $N \to \infty$, the steady-state number of packets moving to the head of free input queues each time slot, and destined for output i, (A^i) becomes Poisson at rate ρ_0 (see Appendix A). These observations and (23) together imply that we can use the results of Section II to obtain an expression for the mean steady-state value of B^i as $N \to \infty$. Modifying (8), we have

$$\overline{B^i} = \frac{\rho_0^2}{2(1-\rho_0)}. \tag{27}$$

However, using (25) and $\bar{F}/N = \rho_0$, we also have

$$\overline{B^i} = 1 - \rho_0. \tag{28}$$

It follows from (27) and (28) that $\rho_0 = (2 - \sqrt{2}) = 0.586$ when the switch is saturated and $N = \infty$.

It is interesting to note that this same asymptotic saturation throughput has also been obtained in an entirely different context. Consider the problem of memory interference in synchronous multiprocessor systems [11], [12] in which M memories are shared by N processors. Memory requests are presented at the beginning of memory cycles; a conflict occurs if more than one simultaneous request is made to a particular memory. In the event of a conflict, one request is accepted, and the other requests are held for future memory cycles. If $M = N$ and processors always make a new memory request in the cycle immediately following their own satisfied request, then our saturation model for input queueing is identical to the multiprocessor model. As $N \to \infty$, the expected number of busy memories (per cycle) is $(2 - \sqrt{2}) \cdot N$ [11].

When the input queues are saturated and $N < \infty$, the switch throughput is found by analyzing a Markov chain model. Under saturation, the model is identical to the Markov chain analysis of memory interference in [12]. Unfortunately, the number of states grows exponentially with N, making the model useful only for small N. The results presented in Table I,[4] however, illustrate the rapid convergence to the asymptotic throughput of 0.586. In addition, saturation throughputs obtained by simulation[5] (Fig. 6) agree with the analysis.

B. Increasing the Switch Throughput by Dropping Packets

Whenever k packets are addressed for a particular output in a time slot, only one can be transmitted over the output trunk. We have been assuming that the remaining $k-1$ packets wait in their input queues until the next time slot when a new selection is made among the packets that are then waiting. Unfortunately, a packet that could traverse the switch to an idle output during the current time slot may have to wait in queue behind a packet whose output is currently busy. As shown in Section III-A, input queue blocking limits the switch throughput to approximately 0.586 for large N.

Instead of storing the remaining $k-1$ packets in input queues, suppose we just drop them from the switch (i.e., we eliminate the input queues). Dropping packets obviously reduces the switch throughput when the input trunk utilization p is small; more time slots on the output trunks are empty because new packets do not arrive fast enough to replace dropped packets. Although dropping a significant number of packets (say, more than 1 out of 1000) may not be realistic for a packet switch, it is interesting to note that as the input utilization p increases, the reduction in input queue blocking when packets are dropped eventually outweighs the loss associated with dropping the packets.

We define A_m^i as the number of packet arrivals during the mth time slot that are addressed for output i. A_m^i has the binomial probabilities

$$\Pr\,[A_m^i = k] = \binom{N}{k} (p/N)^k (1-p/N)^{N-k}$$
$$k = 0,\, 1,\, \cdots,\, N. \tag{29}$$

We also define the indicator function I_m^i as follows:

$$I_m^i = \begin{cases} 1 & \text{if output trunk } i \text{ transmits a packet} \\ & \quad \text{during the } m\text{th time slot} \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

When we drop packets, only those that arrive during the mth time slot have a chance to be transmitted over output trunks during the mth time slot. If they are not selected in the slot in which they arrive, they are dropped. Consequently, for each output trunk i, the random variables I_r^i and $I_s^i (r \neq s)$ are independent and identically distributed, with probabilities

$$\Pr\,[I_m^i = 1] = \Pr\,[A_m^i > 0]$$
$$= 1 - (1 - p/N)^N. \tag{31}$$

[4] The entries in Table I were obtained by normalizing (dividing by N) the values from [12, Table III].

[5] Rather than plot the simulation results as discrete points, the saturation throughputs obtained for N between 2 and 100 are simply connected by straight line segments. No smoothing is done on the data.

TABLE I
THE MAXIMUM THROUGHPUT ACHIEVABLE WITH INPUT QUEUEING

N	Saturation Throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184
∞	0.5858

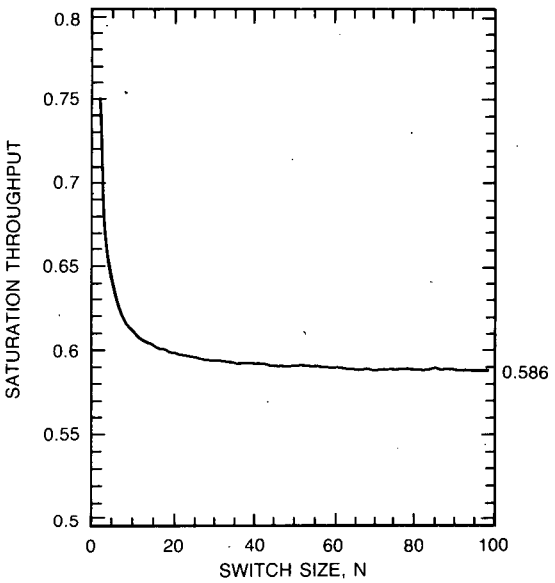

Fig. 6. The maximum throughput achievable with input queueing.

By symmetry, $1 - (1 - p/N)^N$ is the utilization of each output trunk; the switch throughput ρ_0 is given by

$$\rho_0 = 1 - (1 - p/N)^N. \tag{32}$$

As $N \to \infty$,

$$\rho_0 = 1 - e^{-p}. \tag{33}$$

The probability that an arbitrary packet will be dropped from the switch is simply $1 - \rho_0/p$.

The switch throughput ρ_0, as a function of p, is shown in Fig. 7 for several values of N. When the utilization p of the input trunks exceeds a critical threshold, the switch throughput ρ_0 is larger when we drop packets [(32) and (33)] than when we queue them on the input trunks (Table I). For example, when $N = \infty$ and $p > \ln(1 + \sqrt{2})$, the switch throughput when we drop packets is greater than $(2 - \sqrt{2})$ – the throughput with input queues. Table II lists, as a function of p, which of the two strategies yields the larger switch throughput.

C. Waiting Time—Random Selection Policy

Below saturation, packet waiting time is a function of the service discipline the switch uses when two or more input queues are waiting to transmit packets to the same output. In this section, we derive an exact formula for the mean waiting time under the random selection policy for the limiting case of $N = \infty$. The waiting time is obtained by simulation for finite values of N. In Section III-D, numerical results are compared to the mean waiting time under the longest queue and fixed priority selection policies.

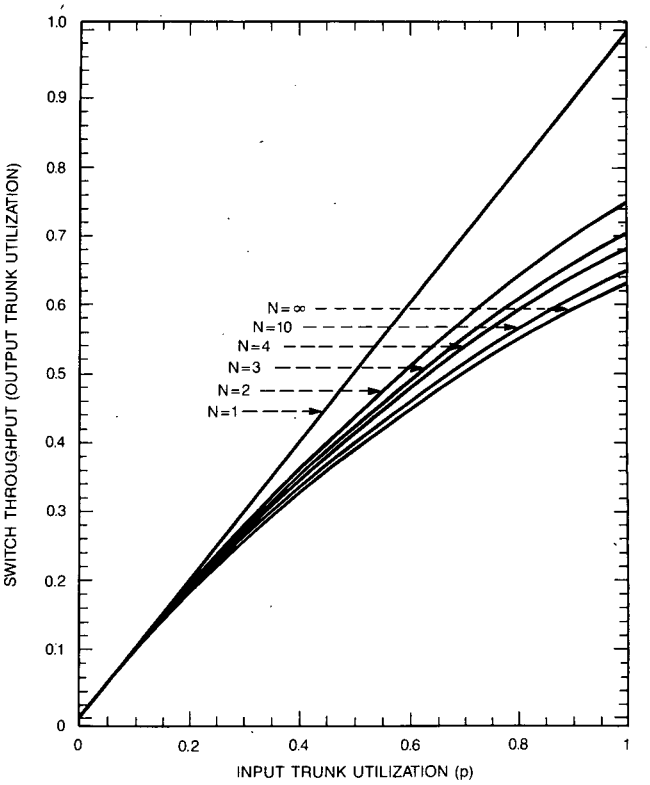

Fig. 7. The switch throughput when packets are dropped, rather than queued at the inputs.

TABLE II
THE STRATEGY (AS A FUNCTION OF p), INPUT QUEUEING, OR PACKET DROPPING THAT YIELDS THE LARGER SWITCH THROUGHPUT

N	Queues On Inputs - Finite Queue Sizes	Queues On Inputs - Saturated Queues	Drop Packets
1	$0 \le p < 1$	$p = 1$	----------
2	$0 \le p < 0.750$	$0.750 \le p \le 1$	----------
3	$0 \le p \le 0.682$	$0.683 \le p \le 0.953$	$0.954 \le p \le 1$
4	$0 \le p \le 0.655$	$0.656 \le p \le 0.935$	$0.936 \le p \le 1$
5	$0 \le p \le 0.639$	$0.640 \le p \le 0.923$	$0.924 \le p \le 1$
6	$0 \le p \le 0.630$	$0.631 \le p \le 0.916$	$0.917 \le p \le 1$
7	$0 \le p \le 0.623$	$0.624 \le p \le 0.911$	$0.912 \le p \le 1$
8	$0 \le p \le 0.618$	$0.619 \le p \le 0.907$	$0.908 \le p \le 1$
∞	$0 \le p \le 0.585$	$0.586 \le p \le 0.881$	$0.882 \le p \le 1$

When the input queues are not saturated, there is a significant difference between our analysis of a packet switch with input queues and the analysis of memory interference in synchronous multiprocessor systems. The multiprocessor application assumes that new memory requests are generated only after a previous request has been satisfied. A processor never has more than one memory request waiting at any time. In our problem, however, packet queueing on the input trunks impacts the switch performance.

A discrete-time Geom/G/1 queueing model (Fig. 8) is used

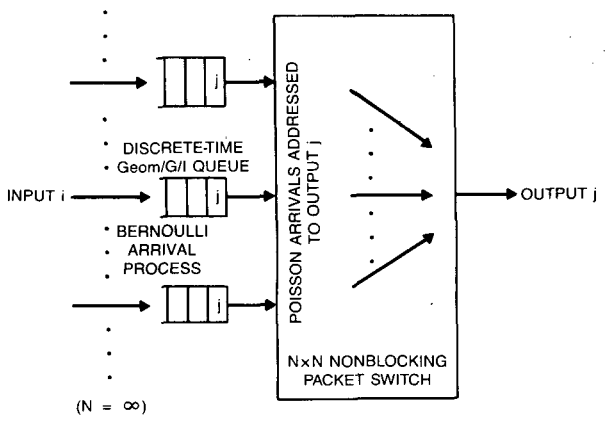

Fig. 8. The discrete-time Geom/G/1 input queueing model used to derive an exact formula for the mean waiting time for the limiting case of $N = \infty$.

to determine the expected packet waiting time for the limiting case of $N = \infty$. The arrival process is Bernoulli: in any given time slot, the probability that a packet will arrive on a particular input is p where $0 < p < 2 - \sqrt{2}$. Each packet has equal probability $1/N$ of being addressed to any given output, and successive packets are independent. To obtain the service distribution, suppose the packet at the head of input queue i is addressed for output j. The "service time" for the packet consists of the wait until it is randomly selected by the switch controller, plus one time slot for its transmission through the switch and onto output trunk j. As $N \to \infty$, successive packets in input queue i experience the same service distribution because their destination addresses are independent and distributed uniformly over all N outputs. Furthermore, the steady-state number of packet "arrivals" to the heads of input queues and addressed for output j becomes Poisson at rate p.[6] Consequently, the service distribution for the discrete-time Geom/G/1 model is itself the packet delay distribution of another queueing system: a discrete-time $M/D/1$ queue with customers served in random order. Analysis of the discrete-time $M/D/1$ queue, with packets served in random order, is given in Appendix B.

Using [6, eq. (39)], the mean packet delay for a discrete-time Geom/G/1 queue is

$$\bar{D} = \frac{p\overline{S(S-1)}}{2(1-p\bar{S})} + \bar{S} \tag{34}$$

where S is a random variable with the service time distribution given in Appendix B and mean value $\bar{S}$. The mean waiting time is $\bar{W} = \bar{D} - 1$

$$\bar{W} = \frac{p\overline{S(S-1)}}{2(1-p\bar{S})} + \bar{S} - 1. \tag{35}$$

$\overline{S(S-1)}$ and $\bar{S}$ are determined numerically using the method in Appendix B.

The mean waiting time $\bar{W}$, as a function of p, is shown in Fig. 9 for both input queueing and output queueing—in the limit as $N \to \infty$. As expected, waiting times are always greater for queueing on inputs than for queueing on outputs. Packet waiting times for input queueing and finite values of N, obtained by simulation,[7] agree with the asymptotic analytic results (Fig. 10).

[6] This follows from the proof in Appendix A.

[7] Rather than plot the simulation results as discrete points, the simulation results are simply connected by straight line segments; no smoothing is done on the data. The same comment applies to Figs. 11, 12, and 13.

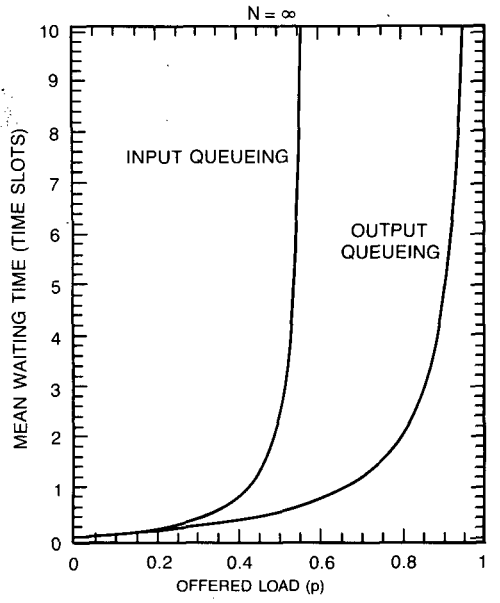

Fig. 9. A comparison of the mean waiting time for input queueing and output queueing for the limiting case of $N = \infty$.

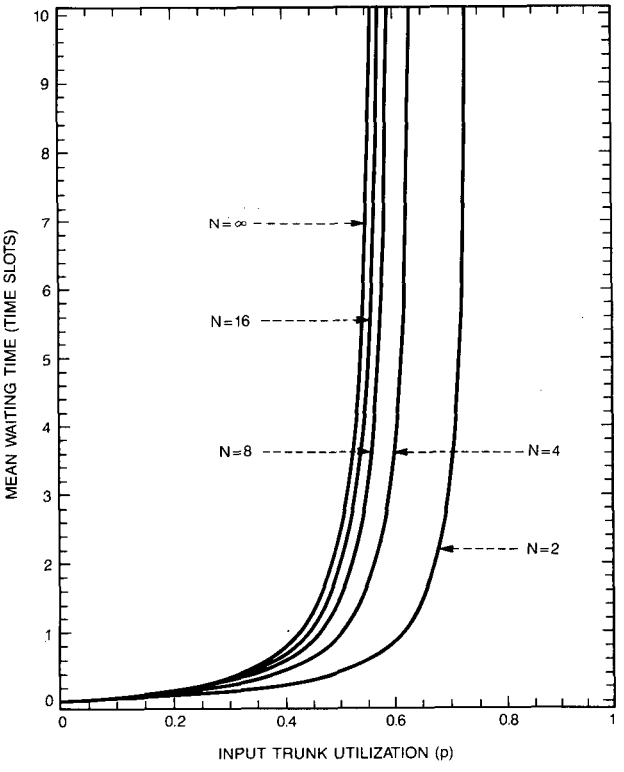

Fig. 10. The mean waiting time for several switch sizes N with input queueing.

D. Longest Queue and Fixed Priority Selection Policies

Until now, we have assumed that if k packets are addressed to a particular output, one of the k packets is chosen at random, each selected with equal probability $1/k$. In this section, we consider two other selection policies: longest queue selection, and fixed priority selection. Under the longest queue selection policy, the controller sends the packet from the longest input queue. A random selection is made if, of the k input queues with packets addressed to a particular output, several queues have the same maximum length. Under the

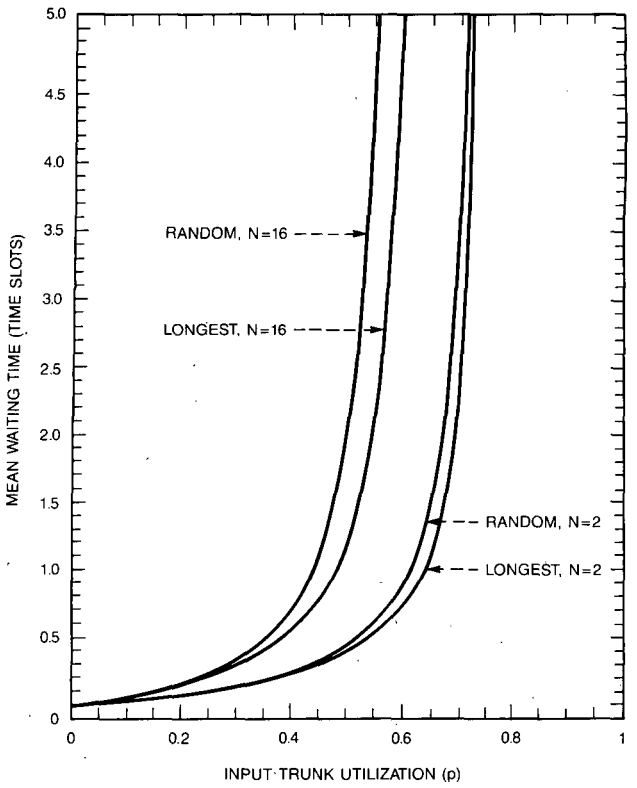

Fig. 11. The mean waiting time for input queueing with the random and longest queue selection policies.

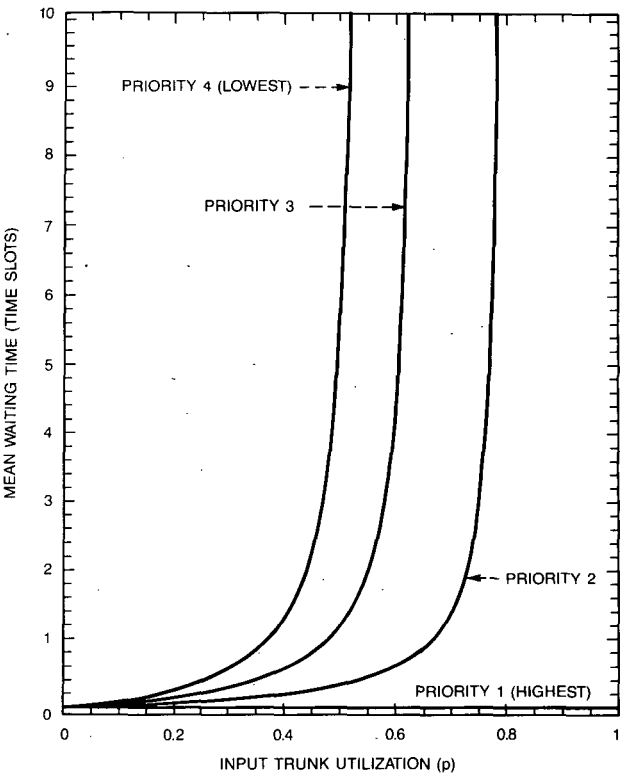

Fig. 12. The mean waiting time for input queueing with the fixed priority service discipline and $N = 4$.

fixed priority selection policy, the N inputs have fixed priority levels, and of the k packets, the controller sends the one with highest priority.

Simulation results for the longest queue policy indicate smaller packet waiting times than those expected with random service (Fig. 11). This is anticipated because the longest queue selection policy reduces the expected number of packets blocked (behind other packets) from traversing the switch to idle outputs.

For the fixed priority service discipline, our simulation results show that the lowest priority input queue suffers large delays and is sometimes saturated, even when the other two service disciplines guarantee stability. Although the saturation throughput is 0.6553 under the random selection policy when $N = 4$ (see Table I), it is shown in Fig. 12 that the lowest priority input queue saturates at approximately 0.55 under the fixed priority discipline. Fig. 13 illustrates the family of waiting time curves for $N = 8$.

These results are interesting because imposing a priority scheme on a single server queueing system usually does not affect its stability; the system remains work conserving. For the $N \times N$ packet switch, however, packet blocking at the low priority input queues does impact stability. More work remains to characterize the stability region.

IV. Conclusion

Using Markov chain models, queueing theory, and simulation, we have presented a thorough comparison of input versus output queueing on an $N \times N$ nonblocking space-division packet switch. What the present exercise has done, for a particular solvable example, is to quantify the intuition that better performance results with output queueing than with input queueing. Besides performance, of course, there are other issues, such as switch implementation, that must be considered in designing a space-division packet switch. The Knockout Switch [13] is an example of a space-division packet switch that places all buffers for queueing packets at the

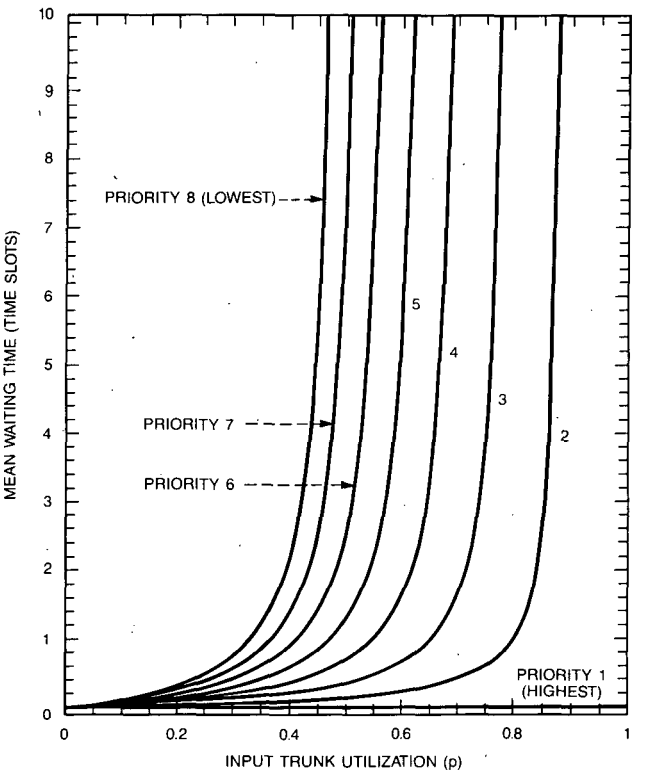

Fig. 13. The mean waiting time for input queueing with the fixed priority service discipline and $N = 8$.

outputs of the switch, thus enjoying the performance advantages of output queueing. Furthermore, the switch fabric runs at the same speed as the input and output trunks.

Appendix A

Poisson Limit of Packets Moving to the Head of Free Input Queues

For the input queueing saturation analysis presented in Section III-A, as $N \to \infty$, we show that the steady-state number of packets moving to the head of free input queues each time slot, and destined for output i, (A^i) becomes Poisson at rate $\rho_0 = \bar{F}/N$. To make clear the dependency of F and ρ_0 on the number of inputs N, we define $\bar{F}(N)$ as the steady-state number of free input queues and $\rho_0(N)$ as the output trunk utilization for a given N where $\rho_0 N = \bar{F}(N)/N$.

We can write

$$\begin{aligned}\operatorname{Var}\left\{\frac{F(N)}{N}\right\} = \frac{1}{N}\Pr[\text{input queue } r \text{ is free}] \\ + \left(1-\frac{1}{N}\right)\Pr[\text{input queue } r \text{ is free, input queue } s (s \neq r) \text{ is free}] \\ - (\Pr[\text{input queue } r \text{ is free}])^2. \end{aligned} \tag{A1}$$

As $N \to \infty$, the events {input queue r is free} and {input queue s is free} become independent for $s \neq r$. Therefore, from (A1),

$$\lim_{N\to\infty} \operatorname{Var}\left\{\frac{F(N)}{N}\right\} = 0. \tag{A2}$$

Given $\epsilon > 0$, we define the set S_N by

$$S_N \triangleq \{L, L+1, \cdots, U-1, U\} \tag{A3}$$

where

$$L \triangleq \max\{1, \lfloor \bar{F}(N) - \epsilon N \rfloor\}, \tag{A4}$$

$$U \triangleq \min\{N, \lceil \bar{F}(N) + \epsilon N \rceil\}, \tag{A5}$$

and $\lfloor x \rfloor$ ($\lceil x \rceil$) denotes the greatest (smallest) integer less than (greater than) or equal to x. By the Chebyshev inequality,

$$\Pr[F(N) \in S_N] \geq 1 - \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}. \tag{A6}$$

Therefore,

$$\begin{aligned}\Pr[A^i = a] &= \sum_{f=\max(a,1)}^{N} \Pr[F(N)=f] \cdot \binom{f}{a}(1/N)^a(1-1/N)^{f-a} \\ &\leq \sum_{f\in S_N} \Pr[F(N)=f] \cdot \binom{f}{a}(1/N)^a(1-1/N)^{f-a} + \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}. \end{aligned} \tag{A7}$$

Case I ($a = 0$): If $f \in S_N$, then

$$(1-1/N)^U \leq \binom{f}{0}(1/N)^0(1-1/N)^{f-0} \leq (1-1/N)^L. \tag{A8}$$

Therefore, from (A6), (A7), and (A8), we obtain

$$\begin{aligned}\left[1 - \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}\right] \cdot (1-1/N)^U \leq \Pr[A^i=0] \\ \leq (1-1/N)^L + \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}. \end{aligned} \tag{A9}$$

As $N \to \infty$,

$$e^{-(\rho_0+\epsilon)} \leq \Pr[A^i=0] \leq e^{-(\rho_0-\epsilon)}. \tag{A10}$$

Since this holds for arbitrarily small $\epsilon > 0$,

$$\lim_{N\to\infty} \Pr[A^i=0] = e^{-\rho_0}. \tag{A11}$$

Case II ($a > 0$): Since $\binom{f}{a}(1/N)^a(1-1/N)^{f-a}$ is a nondecreasing function of f for $1 \leq f \leq N$ and $a > 0$, for $f \in S_N$ we have

$$\begin{aligned}\binom{L}{a}(1/N)^a(1-1/N)^{L-a} &\leq \binom{f}{a}(1/N)^a(1-1/N)^{f-a} \\ &\leq \binom{U}{a}(1/N)^a(1-1/N)^{U-a}. \end{aligned} \tag{A12}$$

Therefore, from (A6), (A7), and (A12), we obtain

$$\begin{aligned}\left[1 - \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}\right] \cdot \binom{L}{a}(1/N)^a(1-1/N)^{L-a} \\ \leq \Pr[A^i=a] \leq \binom{U}{a}(1/N)^a(1-1/N)^{U-a} \\ + \frac{\operatorname{Var}\left\{\frac{F(N)}{N}\right\}}{\epsilon^2}. \end{aligned} \tag{A13}$$

As $N \to \infty$,

$$e^{-(\rho_0-\epsilon)}\frac{(\rho_0-\epsilon)^a}{a!} \leq \Pr[A^i=a] \leq e^{-(\rho_0+\epsilon)}\frac{(\rho_0+\epsilon)^a}{a!}. \tag{A14}$$

Since this holds for arbitrarily small $\epsilon > 0$,

$$\lim_{N\to\infty} \Pr[A^i=a] = e^{-\rho_0}\frac{\rho_0^a}{a!}. \tag{A15}$$

Appendix B

Discrete-Time $M/D/1$ Queue—Packets Served in Random Order

In this Appendix, we present a simple numerical method for computing the delay distribution of a discrete-time $M/D/1$

queue, with packets served in random order. The number of packet arrivals at the beginning of each time slot is Poisson distributed with rate λ, and each packet requires one time slot for service. We fix our attention on a particular "tagged" packet in the system, during a given time slot. Let $p_{m,k}$ denote the probability, conditioned on there being a total of k packets in the system during the given time slot, that the remaining delay is m time slots until the tagged packet completes service. It is easy to obtain $p_{m,k}$ by recursion on m.

$$p_{1,1} = 1 \tag{B1}$$

$$p_{m,1} = 0 \qquad m \neq 1 \tag{B2}$$

$$p_{1,k} = \frac{1}{k} \qquad k \geq 1 \tag{B3}$$

$$p_{m,k} = \frac{k-1}{k} \cdot \sum_{j=0}^{\infty} p_{m-1,k-1+j} \cdot \frac{e^{-\lambda}\lambda^j}{j!} \qquad m > 1,\ k > 1. \tag{B4}$$

Averaging over k, the packet delay D has the probabilities

$$\Pr[D = m] = \sum_{k=1}^{\infty} p_{m,k} \cdot \Pr[k \text{ packets in system immediately after the tagged packet arrives}]$$

$$= \sum_{k=1}^{\infty} p_{m,k} \cdot \sum_{n=0}^{k-1} q_n \cdot \frac{e^{-\lambda}\lambda^{k-n-1}}{(k-n-1)!} \tag{B5}$$

where the q_n are the steady-state queue size probabilities given by (13)–(15).

The variance and mean of the packet delay distribution are determined numerically from the delay probabilities in (B5).

References

[1] J. S. Turner and L. F. Wyatt, "A packet network architecture for integrated services," in *GLOBECOM'83 Conf. Rec.*, Nov. 1983, pp. 45–50.

[2] J. J. Kulzer and W. A. Montgomery, "Statistical switching architectures for future services," in *Proc. Int. Switching Symp.*, May 1984.

[3] T.-Y. Feng, "A survey of interconnection networks," *Computer*, vol. 14, pp. 12–27, Dec. 1981.

[4] D. M. Dias and M. Kumar, "Packet switching in N log N multistage networks," in *GLOBECOM'84 Conf. Rec.*, Nov. 1984, pp. 114–120.

[5] Y.-C. Jenq, "Performance analysis of a packet switch based on a single-buffered Banyan network," *IEEE J. Select. Areas Commun.*, vol. SAC-1, pp. 1014–1021, Dec. 1983.

[6] T. Meisling, "Discrete-time queueing theory," *Oper. Res.*, vol. 6, pp. 96–105, Jan.–Feb. 1958.

[7] I. Rubin, "Access-control disciplines for multiaccess communication channels: Reservation and TDMA schemes," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 516–536, Sept. 1979.

[8] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*. New York: Wiley, 1975.

[9] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*. New York: Wiley, 1974.

[10] P. J. Burke, "Delays in single-server queues with batch input," *Oper. Res.*, vol. 23, pp. 830–833, July–Aug. 1975.

[11] F. Baskett and A. J. Smith, "Interference in multiprocessor computer systems with interleaved memory," *Commun. ACM*, vol. 19, pp. 327–334, June 1976.

[12] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, pp. 897–908, Sept. 1975.

[13] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," in *Proc. Int. Switching Symp.*, Phoenix, AZ, Mar. 1987, pp. 801–808.

★

Mark J. Karol (S'79–M'85) was born in Jersey City, NJ, on February 28, 1959. He received the B.S. degree in mathematics and the B.S.E.E. degree in 1981 from Case Western Reserve University, Cleveland, OH, and the M.S.E., M.A., and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1982, 1984, and 1985, respectively.

Since 1985 he has been a member of the Network Systems Research Department at AT&T Bell Laboratories, Holmdel, NJ. His current research interests include local and metropolitan area lightwave networks, and wide-band circuit- and packet-switching architectures.

★

Michael G. Hluchyj (S'75–M'82) was born in Erie, PA, on October 23, 1954. He received the B.S.E.E. degree in 1976 from the University of Massachusetts, Amherst, and the S.M., E.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1978, 1978, and 1981, respectively.

From 1977 to 1981 he was a Research Assistant in the Data Communication Networks Group at the M.I.T. Laboratory for Information and Decision Systems where he investigated fundamental problems in packet radio networks and multiple access communications. In 1981 he joined the Technical Staff at Bell Laboratories where he worked on the architectural design and performance analysis of local area networks. In 1984 he transferred to the Network Systems Research Department at AT&T Bell Laboratories, performing fundamental and applied research in the areas of high-performance, integrated communication networks and multiuser lightwave networks. In June 1987 he assumed his current position as Director of Networking Research and Advanced Development at Codex Corporation. His current research interests include wide-band circuit- and packet-switching architectures, integrated voice and data networks, and local area network interconnects.

Dr. Hluchyj is active in the IEEE Communications Society and is a member of the Technical Editorial Board for the IEEE NETWORK Magazine.

★

Samuel P. Morgan (SM'55–F'63) was born in San Diego, CA, on July 14, 1923. He received the B.S., M.S., and Ph.D. degrees, all in physics, from the California Institute of Technology, Pasadena, in 1943, 1944, and 1947, respectively.

He is a Distinguished Member of the Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ. He joined Bell Laboratories in 1947, and for a number of years he was concerned with applications of electromagnetic theory to microwave antennas and to problems of waveguide and coaxial cable transmission. From 1959 to 1967 he was Head, Mathematical Physics Department, and from 1967 to 1982 he was Director, Computing Science Research Center. His current interests include queueing and congestion theory in computer-communication networks.

Dr. Morgan is a member of the American Physical Society, the Society for Industrial and Applied Mathematics, the Association for Computing Machinery, the American Association for the Advancement of Science, and Sigma Xi.