Scheduling Nonuniform Traffic in a Packet-Switching System with Small Propagation Delay

Timothy Weller and Bruce Hajek

***Abstract*—A new model of nonuniform traffic is introduced for a single-hop packet-switching system. This traffic model allows arbitrary traffic streams subject only to a constraint on the number of data packets which can arrive *at* any individual source in the system or *for* any individual destination in the system over time periods of specified length. The nonuniform traffic model is flexible enough to cover integrated data networks carrying diverse classes of data. The system model is rather general, and includes passive optical star wavelength-division networks. Transmission algorithms are introduced for a single-hop packet-switching system with such nonuniform traffic and with propagation delay that is negligible relative to the packet length. The algorithms are based on collision-free scheduling of packets using graph-matching algorithms since the global state of the system is known to all stations at any time. A companion paper introduces transmission algorithms for the same network and traffic model, except with a propagation delay that is very large compared to the packet length.**

I. Introduction

Architects of integrated data networks must deal with diverse traffic types which are nearly impossible to model precisely. This motivates the introduction of traffic models based on *constraints* to be satisfied by traffic streams. The goal of this paper is to study methods for transmitting nonuniform traffic in a simple packet-switching system. A new model of nonuniform traffic is introduced which is based on traffic constraints. Most previous related work (see [1] for an overview) has focused on uniform traffic patterns. Traditional approaches such as time-division multiplexing and pure random contention schemes are not well suited for nonuniform traffic. At least some adaptive reservation of bandwidth is generally required.

The basic model of a packet-switching system and the model of nonuniform traffic are presented in the remainder of this section. In Section II, transmission algorithms are considered in which packets are collected into batches before transmission scheduling, while in Section III transmission algorithms are considered which continuously schedule packets for transmission as needed. The algorithms of Sections II and III are based on several assumptions, the following three of which are examined in Section IV: the availability of a common control channel, the availability of one data channel per transmitter, and the ability of transmitters to access all virtual source queues within their queue.

Manuscript received July 1, 1996; revised July 26, 1997; approved by IEEE/ACM Transactions on Networking Editor R. Ramaswami. This work was supported by the National Science Foundation under Contract NSF NCR 9004355 and Contract NSF NCR 93-14253 and by an AT&T Graduate Fellowship. An earlier version of this work was presented at IEEE INFOCOM'94.

T. Weller was with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. He is now with Donaldson, Lufkin & Jenrette, New York, NY 10172 USA (e-mail: tweller@dlj.com).

B. Hajek is with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: b-hajek@uiuc.edu).

Publisher Item Identifier S 1063-6692(97)08780-3.

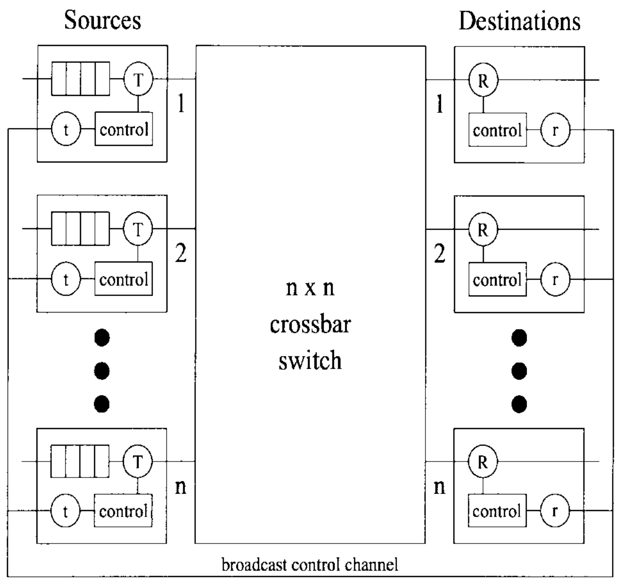

Fig. 1. Basic model.

A. Overview of the Basic Switching System Model

The *basic model* of a packet-switching system considered in this paper has many stations with a fully connected topology (see Fig. 1). This multiple-access communication model was used previously to analyze optical networks [2], [3], but its application can be broader. Each station is both a source and destination[1] and each has a data transmitter and receiver. The fully connected topology is realized by dedicating a data channel to each source, and allowing a receiver to receive from any data channel. Time is slotted. Data are transmitted synchronously in fixed-size packets. All data channels have a common slot length. Each source can transmit at most one packet during a given slot, and each destination can receive at most one packet during a given slot.

There exists a low-bandwidth broadcast control channel available to all stations for the exchange of source queue

[1] A station may send a packet to itself in this model. Removing this assumption requires only minor modifications of the transmission algorithms.

state information and future transmission information. This channel might, for example, share the physical medium with the data channels. The control channel bandwidth should be a small fraction of the total bandwidth. Each station has a separate transmitter and receiver for the control channel. Unless otherwise noted, the control channel is shared using time-division multiplexing, with each station allowed to transmit during one control channel minislot in each data slot. A *transmission algorithm* produces a *slot assignment*, which maps each transmission slot on each data channel to a particular virtual queue (source–destination pair). An example of a communication system which fits the basic model is a broadcast and select network using a passive optical star as in the RAINBOW project at IBM [3]. The data transmission occurs using wavelength-division multiplexing. Each transmitter sends data on a fixed unique wavelength, and each receiver has a tunable filter to receive from any one particular transmitter per slot. The control channel is a single unique wavelength, shared among the stations using time-division multiplexing. Each station needs a fixed transmitter and a fixed receiver on this wavelength. RAINBOW is designed primarily for use in metropolitan-area networks. A similar broadcast and select network is described in [4]. The basic model can also be applied to large all-electronic switches with input buffering.

Each source has a buffer to hold the queue of packets which have not yet been sent. More than one packet can arrive at a source queue during a single slot. Each source queue can be viewed as many separate *virtual queues* by partitioning traffic by destination. The propagation delay between stations is assumed to be negligible in this paper. In the companion paper [5], the same network and traffic model are considered, but for the case of very large propagation delay.

Since the propagation delay between stations is negligible and the control channel has sufficient capacity, each station can be made immediately aware of arrivals at all other stations. Collision-free scheduling is possible because the state of each virtual queue is globally available if each station executes a known transmission algorithm. The scheduling algorithms of this paper are also useful for systems with very low d_{prop}, such as a packet switch with a central controller. For any nonzero d_{prop}, the transmission algorithms of this paper can be used with an additional delay of d_{prop} for each packet. Each packet is simply ignored by the transmission algorithm for d_{prop} slots after its arrival while notification of its arrival propagates to all stations.

While the analysis of scheduling algorithms given in this paper is based on deterministically constrained traffic, similar algorithms have been shown to have good performance for random traffic. For example, [6] considers an algorithm in the class CONTINUOUS_STATIC defined in this paper, and shows good performance for traffic that is independent and identically distributed in time and possibly asymmetrical with respect to destination nodes. Also, interesting stability results have been proved in [7, Proposition 3] and [8] for random traffic and scheduling algorithms based on weighted maximum matchings.

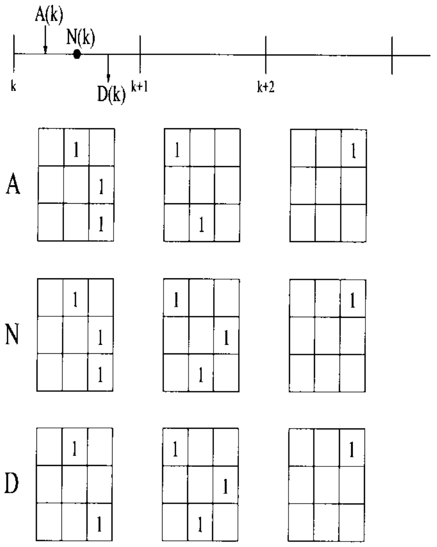

Fig. 2. Matrix sequences.

B. Notation and Assumptions

Let $\mathcal{N} = \{1, \cdots, n\}$ index the set of stations. The virtual queues of source i are labeled (i, j) for $j \in \mathcal{N}$. Slot k refers to the time period $[k, k+1)$. The sequence of consecutive integers (slots) $k, \cdots, l-1$ is written $[k: l)$. Let Z be the set of integers, and let Z^+ be the set of nonnegative integers.

During a given slot k, new packets arrive at a source, and are placed in the buffer at the beginning of the slot. Next, the source makes a decision about which (if any) packet to send. Any packet in the buffer is eligible for transmission. Finally, the selected packet is transmitted, and is said to depart. The *access delay* of a packet arriving in slot k and departing in slot j is defined to be $j-k$. Because a packet can arrive and depart in the same slot, the minimum possible access delay is zero.

Fix an arrival sequence. For any packet p in the sequence, let d_p denote the access delay of p. Let $d_{\max} = \max_p (d_p)$ denote the maximum access delay for the sequence, and let d_{ave} denote the average access delay for the sequence, defined by $d_{\mathrm{ave}} = \limsup_{T\to\infty} d_{\mathrm{ave}}(T)$ where $d_{\mathrm{ave}}(T) = (1/a_T) \Sigma_{p\in\mathcal{A}_T} d_p$, $\mathcal{A}_T$ denotes the set of packets that arrive by time T, and $a_T = |\mathcal{A}_T|$, the cardinality of $\mathcal{A}_T$, for $T \in Z^+$.

The notation for arrivals, backlogs, and departures during slot k is as follows. The number of arrivals at source i destined for j is $A_{ij}(k)$. *After* the arrivals but *before* departures, the number of packets in virtual queue (i,j) is $N_{ij}(k)$. The number of departures from source i to destination j is $D_{ij}(k)$, which is either 0 or 1. Because there are n sources and n destinations, these elements can be represented collectively as three $n \times n$ matrices: the arrival matrix $\boldsymbol{A}(k)$, the backlog matrix $\boldsymbol{N}(k)$, and the departure matrix $\boldsymbol{D}(k)$ (see Fig. 2). With this matrix representation, packets that arrive at (depart from) source i for destination j are said to arrive at (depart from) cell (i, j), row i, and column j. Let $R_i(k) = \Sigma_{j=1}^n N_{ij}(k)$ be the number of packets in row i of the backlog in slot k. Let $C_j(k) = \Sigma_{i=1}^n N_{ij}(k)$ be the number of packets in column j of the backlog in slot k.

Any matrix with nonnegative integer entries is called a *traffic matrix* ($\boldsymbol{A}(k)$, $\boldsymbol{N}(k)$, and $\boldsymbol{D}(k)$ are traffic matrices for every k). The arrival matrix sequence $\boldsymbol{A} = (\boldsymbol{A}(k))_{k \in Z}$ describes the entire arrival process of all packets for all sources and destinations. The system has no arrivals or departures until slot 0. The initial backlog matrix $\boldsymbol{N}(-1)$ and the departure matrix $\boldsymbol{D}(-1)$ are taken to be the matrix of all zeros, unless otherwise noted. The backlog evolution is very simple, and is given by $\boldsymbol{N}(k) = \boldsymbol{N}(k-1) - \boldsymbol{D}(k-1) + \boldsymbol{A}(k)$ for $k \geq 0$. The $\boldsymbol{D}(k)$ matrices are *switching matrices*, which means they are 0–1 matrices with at most a single 1 in any row or column. Each switching matrix represents a one-to-one correspondence between a set of sources and a set of destinations. It is assumed that the switching time is zero.

C. A New Nonuniform Traffic Model

Many models of nonuniform traffic streams have been proposed (see, for example, [9]–[11]). These can generally be classified into two categories. *Random* models have random arrivals with a specified distribution that allows for the possibility of nonuniform traffic matrix sequences. Examples include Markov-modulated Poisson arrivals and geometric burst-length on/off arrivals. *Deterministic constraint* models do not require an underlying arrival distribution, but instead require that each traffic matrix sequence meet some specified constraints. It is the second type of model which most easily simultaneously allows significant nonuniformity of traffic and worst case analysis for quality of service. The class of deterministic constraint models introduced in this paper is called *line-constrained* nonuniform traffic models.

The term *line* is used to refer to either a row or column of a matrix. A *line sum* is the sum of all the elements in a line of a matrix. The maximum line sum of a matrix is indicated by the operator $\|\cdot\|$. *Line backlogs* refer to the line sums of the matrices in $\boldsymbol{N}$, and the *maximum line backlog* is $\max_{k \in Z} \|\boldsymbol{N}(k)\|$, denoted $\|\boldsymbol{N}\|$. Each line sum of the arrival matrix sequence $\boldsymbol{A}$ can be viewed as a traffic stream. Note that a given packet is an element of two streams—one corresponding to its row (source) and one corresponding to its column (destination). There are $2n$ traffic streams in a traffic matrix sequence.

Line-constrained traffic models constrain each line (each traffic stream) of the arrival matrix sequence. By constraining each of the $2n$ traffic streams, the arrival matrix sequence can be constrained while allowing for some bursts of data. Many examples of constraints on individual traffic streams exist, such as the (σ, ρ) constraint of [9] and the stochastic domination constraint of [11]. In this paper, an (α, S) constraint is used, defined as follows. A traffic stream is said to be (α, S) if, in any S consecutive slots, the stream contains at most αS packets, where S is a positive integer, and $0 < \alpha \leq 1$. For convenience, *αS is taken to be an integer*. This traffic model is a "continuous-arrival" variant of the notion of partial h relation [12], in which there is a batch of packets to route, at most h starting at any node and at most h destined for any node.

The parameter α represents the maximum long-run throughput allowed for any source or destination. The parameter S is the smallest time period over which the throughput constraint is enforced. An (α, S) traffic sequence can thus be simply described: "over any time period of length S, no more than αS packets arrive at a given source, and no more than αS packets bound for a given destination arrive at all sources." A random sequence is said to be (α, S) if its sample paths meet the (α, S) constraint. Arrival sequences with sources exhibiting on/off bursts or "hot spots"—commonly studied as representative nonuniform traffic—are easily accommodated with the (α, S) model. Periodic, multiplexed traffic is easily accommodated by the (α, S) model.

With two degrees of freedom provided by the choice of α and S, the (α, S) traffic model is a flexible and widely applicable model of nonuniform traffic. *The design and analysis of transmission algorithms for (α, S) traffic are the focus of this paper.*

D. The Time Slot Assignment Problem

The most direct approach to global information scheduling for (α, S) traffic borrows from work on a classical problem in satellite switching. It is worth a brief digression to examine this problem, known as the time slot assignment problem (TSA) for an SS/TDMA system. An *assignment* for a traffic matrix is a finite sequence of switching matrices whose sum equals the traffic matrix. The *duration* of the assignment is the length of the sequence. The number of *modes* of the assignment is the number of unique switching matrices in the sequence.

The TSA problem [13] can be stated simply: given a traffic matrix, find an assignment of minimum duration which also minimizes the number of switching modes. This problem has been shown to be NP complete [14]. For small (or zero) switching times, the goal is simply to find a minimum duration assignment. In the packet-switching context, it is also desirable to find assignments which minimize the delay of packets in the traffic matrix. Several scheduling algorithms for (α, S) traffic are derived here from the principles behind the solution of the TSA problem for SS/TDMA.

A fundamental result of combinatorial mathematics [15] is that any traffic matrix with maximum line sum m can be represented as the sum of m switching matrices. Any algorithm which finds such an assignment is said to be 100% *efficient*. It is well known that 100% efficient algorithms exist which work one step at a time, subtracting switching matrices from the remaining traffic matrix (see, for example, [13]). Any matrix has an associated bipartite graph (called by the same name as the matrix) which is constructed as follows. Let one set of nodes correspond to rows of the matrix, and the other set of nodes correspond to columns. Let there be an edge between any two nodes where the corresponding matrix position has a nonzero element. A node is *critical* if the corresponding line in the matrix has maximum line sum. At each step, the switching matrix to subtract can be found by finding a maximum matching[2] on the traffic graph (associated with the

[2] A matching in a graph is any set of edges such that no two edges share a node. A *maximum* matching is a matching of maximum cardinality. A *maximal* matching is a matching which cannot be extended without removing edges from the matching. For a $2n$-node bipartite graph, maximum matching can be done in time $O(n^{2.5})$ and maximal matching can be done in time $O(n^2)$. Note that a maximum matching is a maximal matching. See [16] for details.

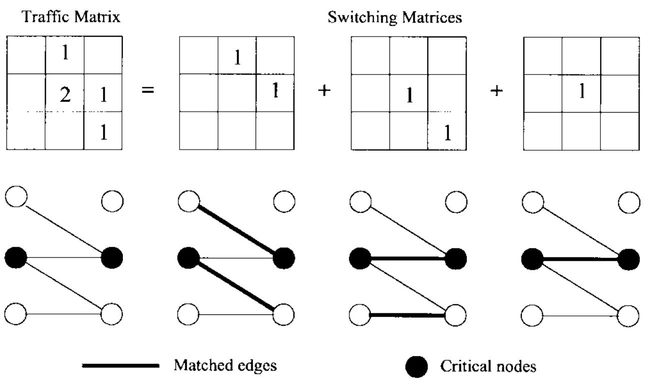

Fig. 3. Time slot assignment example.

remaining traffic matrix) which covers the critical nodes of this graph. This is called a *critical maximum matching*. See Fig. 3 for a TSA example. (Note that there is a maximum matching for the original traffic graph in Fig. 3 which does not cover source 2. Use of this noncritical matching would lead to an assignment of at least four slots.) Fast algorithms presented for edge-coloring bipartite graphs (see [17], for example) can also be used on the TSA problem. The critical maximum matching method for the TSA solution is better suited for continuous operation, as dicussed in Section III.

II. Batch Scheduling

The following properties of network operation are desirable.

P1 Maximum access delay is bounded.
P2 Average access delay is bounded.
P3 Each line backlog is bounded.
P4 Each line backlog hits zero persistently.

A transmission algorithm is said to have property P only if property P is true for all (α, S) arrival sequences in conjunction with the transmission algorithm. A class of transmission algorithms is said to have property P if every algorithm in that class has property P. These properties serve as design goals for the scheduling algorithms developed in this paper. A simple method of insuring properties (P1)–(P3) is to collect packets into batches and apply a TSA algorithm, as described next.

For each $b \geq 0$, let *batch interval* b denote the the interval of slots $[bL\colon (b+1)L)$, where $L \geq 1$ denotes the duration of each batch interval. A batch scheduling algorithm queues the *batch* consisting of all packets that arrive during batch interval b, and schedules this batch of packets during the interval $[(b+1)L-1\colon (b+2)L-1)$ using some algorithm. This departure interval is batch interval $b+1$ moved forward by one slot (some batch b packets can arrive and depart in slot $(b+1)L-1$). The backlog matrix sequence $\boldsymbol{N}$ is related to another sequence $\boldsymbol{N}^B$, called the *batch backlog matrix* sequence, defined as follows. Let $\boldsymbol{N}^B(k), k = 0, 1, \cdots$, be

$$\boldsymbol{N}^B(k) = \begin{cases} \boldsymbol{N}(k), & k = bL-1, b \in Z^+ \\ \boldsymbol{N}^B(k-1) - \boldsymbol{D}(k-1), & \text{otherwise.} \end{cases}$$

Note that $\boldsymbol{N}^B(-1) = \boldsymbol{N}(-1) = 0$ so that $\boldsymbol{N}^B(L-1)$ is the first nonempty batch backlog matrix. The batch backlog matrices differ from the backlog matrices in that arrivals are added to the batch backlog matrices only in slots $L-1,\ 2L-1, \cdots$ so that $\boldsymbol{N}^B(bL-1) = \Sigma_{k=(b-1)L}^{bL-1} \boldsymbol{A}(k)$.

Suppose that $\alpha \leq 1$. Then the packets that arrive in any αS consecutive slots can be scheduled by a TSA algorithm in αS slots, so we take $L = \alpha S$. (We could have taken $L = S$, but the choice $L = \alpha S$ allows us to take advantage of the fact that an (α, S) traffic sequence is also a $(1, \alpha S)$ traffic sequence.) Let BATCH_TSA be the class of scheduling algorithms which in each slot subtract, from the most recent batch backlog matrix, a switching matrix found by a critical maximum matching on the associated batch backlog graph. This switching matrix is the departure matrix.

Theorem 2.1: For $\alpha \leq 1$, BATCH_TSA with duration of batch intervals $L = \alpha S$ has properties (P1)–(P3). The maximum access delay is no more than $2(\alpha S - 1)$. The average access delay is no more than $\frac{3}{2}(\alpha S - 1)$. The maximum line backlog is no larger than $2\alpha S$.

Proof: Fix any scheduling algorithm in BATCH_TSA with $L = \alpha S$ and any (α, S) arrival sequence. The packets that arrive during any batch interval b are scheduled to depart during $[(b+1)\alpha S - 1\colon (b+2)\alpha S - 1)$. Thus, without loss of generality, consider batch 0. Any packet in the batch arrives in slot 0 or later and departs in slot $2(\alpha S - 1)$ or sooner. Therefore, $d_{\max} \leq 2(\alpha S - 1)$. The number of batch 0 packet departures per slot is nonincreasing in the consecutive slots $[\alpha S - 1\colon 2\alpha S - 1)$. Thus, $d_{\text{ave}} \leq \frac{3}{2}(\alpha S - 1)$. At any time, packets from at most two batches may be in the source queues so $||\boldsymbol{N}|| \leq 2\alpha S$. □

An alternative to using critical maximum matchings is to use maximal matchings, which are matchings that are not subsets of larger matchings, or to use arbitrary maximum matchings. Maximal matchings need not have the largest size among all matchings, nor do they need to cover critical nodes, so they generally require much less effort to compute than critical maximum matchings. Let BATCH_MAXIMAL (resp., BATCH_MAXIMUM) be the class of batch scheduling algorithms which in each slot subtract a switching matrix from the most recent batch backlog matrix, with the switching matrix being a maximal (resp., maximum) matching on the associated batch backlog graph. Any scheduling algorithm in BATCH_TSA is in BATCH_MAXIMUM, and any algorithm in BATCH_MAXIMUM is in BATCH_MAXIMAL. The use of arbitrary maximal or maximum matchings requires at most $2\alpha S - 1$ slots to evacuate a single batch of packets with maximum line sum αS since a given packet in the batch can be delayed by at most $2\alpha S - 2$ other packets, which must be in the same row or column. It is thus natural to take the duration L of batch intervals to be $2\alpha S - 1$ for BATCH_MAXIMAL or BATCH_MAXIMUM algorithms.

The following theorem is a positive result for a simple class of scheduling algorithms, BATCH_MAXIMAL.

Theorem 2.2: For $\alpha \leq 1/2$, BATCH_MAXIMAL (and hence also BATCH_MAXIMUM), with duration of batch intervals $L = 2\alpha S - 1$, has properties (P1)–(P3). The maximum access delay is no more than $4(\alpha S - 1)$. The average access delay is no more than $3(\alpha S - 1)$. The maximum line backlog is no larger than $2\alpha S$.

Proof: The bounds for $d_{\max}$ and d_{ave} are established as in the proof of Theorem 2.1 replacing αS by $2\alpha S - 1$. The line backlog bound is the same as in Theorem 2.1 by the same reasoning. □

Remark 2.1: For $\alpha > 1/2$, Appendix B gives an example demonstrating that for n and S sufficiently large, some algorithms in BATCH_MAXIMUM (and hence some algorithms in BATCH_MAXIMAL) do not have any of the properties (P1)–(P3).

The distributed routing algorithm for a packet switch described in [18] provides the inspiration for a class of algorithms introduced here called PARALLEL_MAXIMAL, which is similar to the class BATCH_MAXIMAL. For a given batch, consider an empty scheduling grid with n rows (representing the destinations) and $2\alpha S - 1$ columns (representing the departure slots for this batch). Each source proceeds in parallel independently, monitoring the actions of other sources by use of the common scheduling grid. Consider source i trying to schedule a packet for destination j. The source checks the scheduling grid to find a column which does not contain an "i" in any row and which has an empty cell in row j. Upon finding such a column, it writes "i" in the corresponding cell of row j. Source i then chooses another packet, and continues in the same manner until it has no unassigned packets. A source can always place each packet, by the same reasoning used for BATCH_MAXIMAL. Any strategy may be used by a source. If the sources proceed synchronously through n scheduling stages, and source i tries to schedule all of its packets destined for j during stage $(i+j)$ mod n, the algorithm is roughly equivalent to that of [18]. This strategy is attractive because multiple sources are not simultaneously considering the same destination during a given slot. Other strategies may include randomization, although the lowest average access delay is likely to be obtained by searching through the columns in order of increasing time. If all of the sources step through the time slots (scheduling grid columns) in unison, then the resulting PARALLEL_MAXIMAL algorithm is a BATCH_MAXIMAL algorithm.

BATCH_TSA and BATCH_MAXIMAL algorithms provide a maximum access delay of $2(\alpha S - 1)$ and $4(\alpha S - 1)$, respectively. For nonzero d_{prop}, an additional d_{prop} for global information must be added. If αS is small, then batch scheduling algorithms perform very well for zero (or small) propagation delay in the sense that delay is small and throughput as high as 1 can be obtained. As αS increases, it is natural to ask if better scheduling is possible. The next section addresses this question by allowing packets to be considered for departure immediately upon arrival without waiting for transmission of the previous batch.

III. Continuous Scheduling

The continuous scheduling algorithms of this section find the departure matrices in each slot by using a matching on the backlog graph associated with the backlog matrix in the same slot. No batching occurs. The proofs of the theorems in this section are relegated to Appendix A.

TABLE I
Performance Bounds for Zero Propagation Delay Algorithms

Class of Scheduling Algorithms	Range of Throughput	Property P3 $\|\|N\|\|$	Property P2 d_{ave}	Property P1 d_{max}
BATCH_TSA	$\alpha \le 1$	$2\alpha S$	$\frac{3}{2}(\alpha S - 1)$	$2(\alpha S - 1)$
BATCH_MAXIMAL or	$\alpha \le \frac{1}{2}$	$2\alpha S$	$3(\alpha S - 1)$	$4(\alpha S - 1)$
BATCH_MAXIMUM	$\alpha > \frac{1}{2}$	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆
CONTINUOUS_MAXIMAL or	$\alpha \le \frac{1}{3}$	αS	$2(\alpha S - \frac{1}{2})$	$6(\alpha S - \frac{1}{3})$
CONTINUOUS_MAXIMUM	$\alpha > \frac{1}{2}$	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆
OLDEST_MAXIMAL	$\alpha \le \frac{1}{2}$	αS	$2(\alpha S - 1)$	$2(\alpha S - 1)$
CONTINUOUS_STATIC	$\alpha > \frac{1}{2}$	ooo	ooo	ooo

⋆⋆⋆ Property does not hold for some algorithms in the class.

ooo Property does not hold for any algorithm in the class.

Let CONTINUOUS_MAXIMAL be the class of continuous scheduling algorithms which use an arbitrary maximal matching in each slot. This is perhaps the simplest class of scheduling algorithms presented in this paper. A subset of CONTINUOUS_MAXIMAL is the class CONTINUOUS_MAXIMUM of continuous scheduling algorithms which use an arbitrary maximum matching in each slot.

Theorem 3.1: For $\alpha \le 1/3$, CONTINUOUS_MAXIMAL (and therefore also CONTINUOUS_MAXIMUM) has properties (P1)–(P3). The maximum access delay is no more than $6(\alpha S - 1/3)$. The average access delay is no more than $2(\alpha S - 1/2)$. The maximum backlog is no larger than αS.

Let OLDEST_MAXIMAL be the subclass of CONTINUOUS_MAXIMAL which use any oldest first maximal matching in each slot. An oldest first maximal matching is a matching with the following property: any packet not in the matching is blocked by a packet in the matching which arrived at the system not later than the blocked packet. Such a matching can be found by considering packets in order of arrival, adding each to the matching as long as no conflict is caused.

Theorem 3.2: For $\alpha \le 1/2$, OLDEST_MAXIMAL has properties (P1)–(P3). The maximum access delay is no more than $2(\alpha S - 1)$. The average access delay is no more than $2(\alpha S - 1)$. The maximum line backlog is no larger than αS.

Finally, note that continuous scheduling for $1/2 < \alpha \le 1$ may be possible. However, the matchings used for such scheduling must depend on time, random events, or the system history due to the following negative result. Let CONTINUOUS_STATIC be the class of continuous scheduling algorithms which use a fixed deterministic matching algorithm applied to the backlog graph in each slot. The class CONTINUOUS_STATIC does not include OLDEST_MAXIMAL algorithms, but does include some algorithms in CONTINUOUS_MAXIMUM and CONTINUOUS_MAXIMAL.

Theorem 3.3: For $\alpha > 1/2$, $S \ge 6$, and $n \ge 3$, no scheduling algorithm in the class CONTINUOUS_STATIC has any property (P1)–(P4).

Table I summarizes the throughput and delay characteristics of the scheduling algorithms presented in this and the previous section. The last three columns list the upper bounds for $\|\boldsymbol{N}\|$, d_{ave}, and $d_{\max}$, respectively. See Section V for comments on parameter ranges not covered by the table.

The strongest property is (P4), persistent line backlog emptying. For static scheduling algorithms, (P4) holds for only relatively small throughput values, as shown in the following theorem and the associated remark.

Theorem 3.4: For $\alpha \leq (\sqrt{4S+9} - 3/2S\,(\alpha < (1/\sqrt{S})$ asymptotically in S), CONTINUOUS_MAXIMAL has property (P4) and each line backlog hits zero at least once every S slots.

Remark 3.1: For $\alpha \geq (1/\sqrt{S})$, an example in Appendix B demonstrates that at least some algorithms in CONTINUOUS_MAXIMUM (and therefore some in CONTINUOUS_MAXIMAL) do not have property (P4).

Recall that all of the transmission algorithms of this paper require global information, which is assumed to be provided by the control channel with negligible propagation delay. Even if the propagation delay d_{prop} is not negligible, the scheduling algorithms of this paper can be adapted by delaying packets d_{prop} upon arrival. In the companion paper [5], transmission algorithms are introduced for large d_{prop} and a different approach is pursued. While any of the transmission algorithms for nonuniform traffic can be used over the entire range $d_{\text{prop}} \in [0, \infty)$, hybrid combinations of these transmission algorithms and new algorithms may provide better throughput and delay characteristics for specific networks.

IV. Some Implementation Considerations

The capacity requirements on the control channel, modifications allowing for fewer data channels, and the effect of restricting buffer access to head-of-line positions are considered in this section.

A. Control Channel Capacity

The capacity requirements of the control channel are examined in this section. Suppose the control channel is slotted with slot length equal to the data slot length. Each slot is divided into n minislots, one for each source, so that TDMA is the control channel access protocol.

The adaptive scheduling algorithms, continuous or batch, use the control channel to broadcast packet arrival information among stations every l slots. These algorithms implicitly use $l = 1$ and $l = \alpha S$, respectively. By using a larger l, the control channel capacity needed can be reduced, as indicated next. Of course, an additional *notification delay* of l is added to the access delay.

Let C be the minimum control channel capacity required. For simplicity, suppose l is chosen as a multiple of S. Every l slots, each source must broadcast the desired destination for each of at most αl packets. Because fewer than αl packets may be present, the number of possible states per source is the same as the number of ways to place αl balls into $n+1$ bins. Therefore, the average control channel utilization is

$$C = \frac{n}{l} \log \binom{n+\alpha l}{\alpha l} \text{ bits/slot.}$$

Note that $C \leq \alpha n \lceil \log(n+1) \rceil$, which corresponds to separate broadcast notification of every packet ($\alpha n l$ packets per l slots, with $\lceil \log(n+1) \rceil$ bits/packet).

A numerical example illustrates the relationship between n, l, α, and the length of a data packet (slot). Assume that the control channel has the same capacity as a data channel. Suppose a CONTINUOUS_MAXIMAL algorithm is used on a 100-station system where $\alpha = 1/2$. With $l = 2$, $C = 50 \log 101 = 333$ bits. With $l = 10$ and $l = 100$, $C = 265$, and $C = 134$ bits. Clearly, in this example, C is a lower bound on the data packet length. Consider another example in which the data packet is an ATM cell with length 53 bytes (424 bits). Using a CONTINUOUS_MAXIMUM algorithm with $\alpha = 1$ and $l = 1$ (highest throughput, smallest notification delay), $n \log(n+1) \leq 424$ is required. This means that at most 69 stations can be supported. More stations can be supported if the control channel capacity is increased beyond that of a single data channel.

B. Reducing the Number of Data Channels

The previous sections of this paper assume that n data channels are available, with each source assigned to a unique data channel. If fewer data channels are available, the stations are not fully connected in each slot. Suppose the number of data channels is reduced by a factor of c so that there are n/c data channels available.

The simplest strategy for adapting an existing transmission algorithm to work with fewer channels is to use the original algorithm to produce a single slot assignment every $\lceil c \rceil$ slots, and then partition this assignment into $\lceil c \rceil$ pieces. For this purpose, the sources are partitioned into n/c *source groups*, each of size $\lfloor c \rfloor$ or $\lceil c \rceil$. Each *source group* is assigned a unique data channel for transmissions, and the transmissions of the stations within a source group are scheduled by time-multiplexing the existing transmission algorithm over $\lceil c \rceil$-slot frames. In this way, at most one station from each source group transmits in any one slot. Maximum throughput is reduced by $\lceil c \rceil$ due to the c-fold reduction in the number of data channels. For example, a BATCH_MAXIMAL algorithm requires $\lceil c \rceil (2\alpha S - 1)$ slots for each batch, while a BATCH_TSA algorithm requires $\lceil c \rceil \alpha S$ slots for each batch.

The above strategy is wasteful for some values of c such as $c = 1.2$. By allowing *any* n/c sources to transmit in any slot, this waste can be eliminated. Using a BATCH_TSA algorithm with a time slot assignment scheme in [19], each batch with maximum line sum αS can be scheduled in $\lceil c\alpha S \rceil$ slots, which is optimal.

The PARALLEL_MAXIMAL algorithms, described in Section II, are also easily modified for use with a reduced number of data channels. This provides a class of scheduling algorithms based on maximal matchings which, for moderate values of c, can achieve near the minimum possible number of slots for a batch assignment. This supports the intuitive notion that scheduling is easier for a smaller number of data channels (larger c). The scheduling grid used in the modified algorithms has n rows, each corresponding to a destination, and $\lceil (c+1)\alpha S \rceil - 1$ columns, each corresponding to a time slot. Each source proceeds as before, with sources being filled into the grid, but in addition to each column being free of destination conflicts, each column can have no more than one source from each source group. A given source trying to schedule a packet for a given destination can always find a slot because of the following considerations: at most $\alpha S - 1$ columns can have a destination conflict, and at most $\lceil c \rceil \alpha S - 1$ columns can be covered by sources from the same source group. Note that the $\lceil (c+1)\alpha S \rceil - 1$ slots needed to schedule each batch are fewer than the $\lceil c \rceil (2\alpha S - 1)$ slots needed using the simple strategy first described in this section. Although they use only maximal matchings, the PARALLEL_MAXIMAL algorithms approach the BATCH_TSA algorithms in performance for moderate values of c.

C. Source Buffer Access and Head-of-Line Blocking

The transmission algorithms in this paper are assumed to have access to the entire source buffers. Suppose, instead, that an algorithm can transmit only a packet which is in the first l positions in a source queue. This is a likely result of hardware limitations, especially for the large buffers usually required in high-speed networks. In this section, the most severe case, namely, (FIFO) source buffer access $(l = 1)$, is considered. A problem known as head-of-line (HOL) blocking occurs. Because the transmitter at a given source has access only to the head packet in the queue, other packets cannot depart. Such packets might be able to depart if the source queues are more accessible. The classic result for uniform random traffic is that HOL blocking at the source queues limits the maximum throughput to about 0.58 for any transmission algorithm (see [20], for example). For nonuniform traffic patterns, the HOL blocking problem can be much worse.

To illustrate that the HOL blocking problem with (α, S) traffic can be severe, a simpler, related problem is considered. Suppose there are n stations and a total of n^2 packets. Fix any scheduling algorithm that selects a maximal set of packets to depart in each slot. Assume that this scheduling algorithm can access only the HOL positions in the source buffers, and that *no information* about the contents of other buffer positions is available to the algorithm.

Consider the following arrival sequence (which depends on the scheduling algorithm already fixed). The *order* of arrivals is as follows. The first n packets that arrive at the system are bound for destination 1, the next n packets to arrive are bound for destination 2, etc. The *timing* of arrivals is such that the n packets bound for destination 1 arrive during slot 0, one arriving at each source. Thereafter, until all n^2 packets have arrived, whenever a packet departs from a source, a new packet arrives at the same source in the next slot. Note that the destinations of packets arriving at a given source depend on the scheduling algorithm.

The arrival sequence as stated above replaces HOL packets in a "just in time" fashion in order to keep the HOL positions full. However, because the scheduling algorithm is fixed and does not have access to information about source buffers other than the identity of HOL packets, all packets could actually be in the buffers at the beginning of slot 0, and the departure sequence would remain the same as for the original arrival sequence. Packets moving from the second position to the HOL position in a queue are called HOL arrivals. Using this method, all n^2 packets arrive in a batch during slot 0. See Fig. 4 for example arrival and departure sequences.

The time required to schedule the batch of n^2 packets is analyzed next. The system state just after arrivals in slot k can be summarized by $f_k = (e_k | f_{k,1},\ f_{k,2},\ \cdots)$ where $f_{k,i}$ is the number of HOL packets bound for destination i, and e_k is the number of future arrivals that will be bound for the same destination as the most recent HOL arrival. Note that e_k represents the number of future HOL arrivals that occur before a packet with a new destination is introduced to an HOL position. The number of nonzero $f_{k,i}$ is equal to the number of departures in slot k because a maximal set is transmitted. Roughly speaking, the state evolves with e_k decreasing in each slot until it hits zero, at which time it is reset to n, coinciding with the introduction of packets with the next highest destination number. The nonzero $f_{k,i}$ are decreased by one, and then the $f_{k,i}$ corresponding to arrivals is increased by the number of departures (arrivals). For simplicity, let $n = 6b$ for some integer $b \geq 1$. The system begins in state $f_0 = (0|6b,\ 0,\ 0,\ 0,\ \cdots)$, which corresponds to all of the HOL packets bound for destination 1 with no more HOL arrivals in the future for destination 1. The simple system evolution described above yields the following sequence:

$$
\begin{aligned}
f_0 &= (0|6b,\ 0,\ 0,\ \cdots)\\
f_1 &= (6b-1|6b-1,\ 1,\ 0,\ 0,\ \cdots)\\
f_2 &= (6b-3|6b-2,\ 2,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{3b} &= (1|3b,\ 3b,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{5b} &= (2|b,\ b+1,\ 4b-1,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{6b+1} &= (2b|0,\ 0,\ 3b,\ 3b,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{7b+1} &= (0|0,\ 0,\ 2b,\ 4b,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{9b+1} &= (1|0,\ 0,\ 0,\ 2b,\ 4b,\ 0,\ 0,\ \cdots)\\
&\ \vdots\\
f_{11b+2} &= (0|0,\ 0,\ 0,\ 0,\ 2b,\ 4b,\ 0,\ 0,\ \cdots)\\
&\ \vdots
\end{aligned}
$$

Note that f_{11b+2} is a shifted version of f_{7b+1}. Thus, by time $7b+1$, a limit cycle is reached. At the beginning of each cycle, there are two destinations represented by HOL packets [(2/3 of one, 1/3 of the other)], and a new destination is about to be introduced. The average number of departures per slot over a cycle is equal to $12b/(4b+1)$ which is less than 3. Before steady state $(k < 7b+1)$ and after steady state (as the n^2 packet arrivals are exhausted), the average departure rate is also less than 3. Therefore, scheduling the batch of packets requires at least $n^2/3$ slots. Note that all HOL positions in the queues remain full until all n^2 packets reach HOL positions.

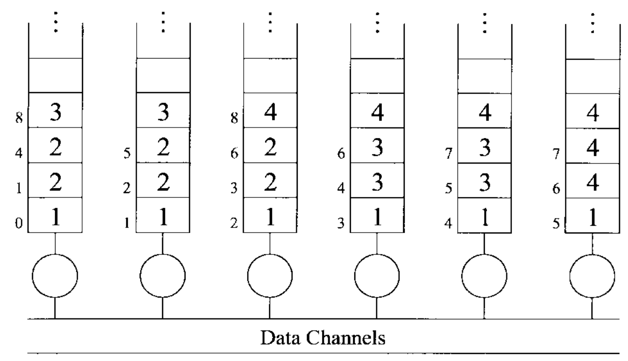

Fig. 4. FIFO source buffers pathological example. The numbers in the buffer cells are packet destinations while the small numbers to the left give the slot in which that packet departs. Note that a maximal set departs in each slot.

Thus, the large evacuation time is due to HOL conflicts, and not due to a lack of HOL packets per se.

Finally, suppose there is an additional system constraint that at most n packets can arrive at (depart from) any source. Modify the arrival sequence so that HOL packets are not replaced at any source after the source has n departures. Then for any time k, fewer arrivals occur during [0: k) than for the original system. To prove this, argue by induction on k that the number of departures for each destination is no larger in the new system than in the original system. Therefore, *for any deterministic scheduling algorithm that can access only HOL positions, there is a batch of n^2 packets, with n packets arriving at each source and n packets bound for each destination, which requires at least $n^2/3$ slots for evacuation.* This indicates that, in the worst case, limited access to source buffers can severely limit throughput for nonuniform traffic in case n and αS are large.

The idea of the above example is to consider arrival streams such that packets with the same destination show up at HOL positions at as nearly the same time as possible. The idea can be applied to other scenarios, such as the basic model of this paper with (α, S) traffic, as we briefly describe next. Suppose α and S are fixed, that n is very large, and that some scheduling algorithm is used for (α, S) traffic which has access (both for purposes of transmission and information) to only HOL packets. Then packets due for each destination can appear at HOL positions in groups of size αS, and the groups then decrease in size by one per slot. This leads to an average of $(\alpha S+1)/2$ HOL positions taken up per distinct destination appearing in at least one HOL position, so the total throughput is limited to $2n/(\alpha S+1)$. This requires that $\alpha \leq 2/(\alpha S+1)$. For example, if $S=100$ and n is sufficiently large, then unless $\alpha \leq 0.13$, no algorithm with access to only HOL positions can keep the backlog bounded.

V. Summary and Future Work

The primary purpose of this paper is to introduce a new model of nonuniform traffic, and to discuss how to transmit this traffic in a packet-switching system. It is hoped that this work is useful to the designers of integrated data networks in the traffic modeling phase and in the transmission algorithm design phase. The new model, namely, (α, S) traffic, imposes deterministic constraints on the joint arrival sequences, whereas the more traditional uniform random traffic model has a stochastic description. It is shown that scheduling (α, S) traffic in a system with small propagation delay is conceptually rather simple. Many variations of matching-based transmission algorithms are available with a range of throughput–delay characteristics.

An open problem is to identify other flexible and useful nonuniform traffic models for which matching-based scheduling algorithms provide high throughput and low access delay. Perhaps implementation and study of such algorithms on existing system installations is called for if accurate models are not available.

At least three open problems for (α, S) traffic remain. First, the conversion of arbitrary traffic sequences to (α, S) traffic sequences is desirable. A multidimensional leaky-bucket-type regulator can be used for each source input stream,[3] but the destination streams (columns of $\boldsymbol{A}$) are not easily regulated due to possible physical separation of the destinations. Optimally dropping packets—in the sense of minimizing the number of dropped packets—which are in violation of the (α, S) constraint prior to system arrival, is conjectured here to be as difficult as optimally scheduling (α, S) traffic. Of course, throwing away excess packets, perhaps chosen by matchings, probably yields a near-minimum number of dropped packets. With current knowledge, the (α, S) traffic model is best used in a system such as in [22] where virtual circuits (between a source–destination pair) reserve bandwidth in advance, respecting the (α, S) constraint. Such connection-oriented traffic is most easily supported, but datagram traffic which is (α, S) is also supported.

The second apparent area for further study of (α, S) traffic is on the "α frontier." For α in the range $1/2<\alpha\leq 1$, it may be possible to produce simple algorithms of a continuous (rather than batch) nature with strong performance guarantees. Theorem 3.3 shows that such algorithms cannot schedule transmissions in each slot as a function of only which source–destination pairs have at least one packet waiting. Perhaps a method based on maximum *weighted* matchings, as proposed for random traffic in [7], [8], would be fruitful. Given the fact that BATCH_MAXIMUM algorithms can achieve $\alpha = 1$ with good performance bounds, it may not warrant much additional effort, unless a very simple algorithm is found. For α in the range $1/3<\alpha\leq 1/2$, we leave two questions open. Do there exist algorithms in CONTINUOUS_STATIC which provide performance properties (P1)–(P3) for this range of α? Do all algorithms in CONTINUOUS_MAXIMAL provide performance properties (P1)–(P3) for this range of α? A "yes" answer to the second question would imply a "yes" answer to the first.

Finally, this work on (α, S) traffic could be extended to a multihop system in which each hop is through a packet switch based on the basic model. Virtual circuits passing through a particular switch could reserve the necessary fraction of the

[3] A leaky bucket regulator (see [21, p. 245]) for a single traffic stream has a one-dimensional state, while each source regulator for a single (α, S) stream requires an S-dimensional state to remember arrivals in S consecutive slots in the past.

"αS packets per S slots" bandwidth from both the switch source and the switch destination during circuit setup. In a network setting, the generalization with different $(\alpha,\ S)$ for each line is probably desirable.

The $(\alpha,\ S)$ traffic model is simple, yet flexible. It is widely applicable, and facilitates the design and analysis of efficient transmission algorithms. Generalization of the model is possible in several directions. In all, the $(\alpha,\ S)$ traffic model should satisfy the network user and system designer while balancing the tradeoffs between their objectives.

APPENDIX A
PROOFS OF THEOREMS ON CONTINUOUS SCHEDULING

The following lemma relates line backlogs and average access delay. It is not immediate from Little's law because no arrival rate is specified.

Lemma A.1: Suppose $d_{\max}$ is finite. For a continuous scheduling algorithm using maximum matchings, $d_{\text{ave}} \leq \|\boldsymbol{N}\|-1$. For a continuous scheduling algorithm using maximal matchings, $d_{\text{ave}} \leq 2\|\boldsymbol{N}\| - 1$.

Proof: In the notation of this paper, the König–Egerváry theorem [23] states that the cardinality of a maximum matching on a backlog graph is equal to the minimum number of lines in the associated backlog matrix which together contain all of the nonzero entries. Let $N(k)$ and $D(k)$ denote the sums of all elements in the matrices $\boldsymbol{N}(k)$ and $\boldsymbol{D}(k)$, respectively. Because no line sum of $\boldsymbol{N}(k)$ is greater than $\|\boldsymbol{N}(k)\|$, the following relationship holds:

$$D(k) \geq \frac{N(K)}{\|\boldsymbol{N}(k)\|} \geq \frac{N(k)}{\max_{k\in Z} \|\boldsymbol{N}(k)\|} = \frac{N(k)}{\|\boldsymbol{N}\|}. \tag{A1}$$

Recall that $\mathcal{A}_T$ is the set of packets that arrived during slots $[0\colon T)$ and $a_T = |\mathcal{A}_T|$. Let d_T be the number of departures during slots $[0\colon T)$. Clearly, $d_T \leq a_T$ because the system starts empty. Let η_T be the total delay suffered in the interval $[T, \infty)$ by all packets in $\mathcal{A}_T$. Let $\tilde{\eta}_T$ be the total delay suffered in the interval $[T, \infty)$ by all packets in $\mathcal{A}_T$. The following development is similar to the development of Little's law. The second inequality uses (A1).

$$\begin{aligned} d_{\text{ave}}(T) &= \frac{\eta_T + \tilde{\eta}_T}{a_T} \leq \frac{\eta_T}{d_T} + \frac{\tilde{\eta}_T}{a_T} \\ &= \frac{\sum_{k=0}^{T-1}[N(k) - D(k)]}{\sum_{k=0}^{T-1} D(k)} + \frac{\tilde{\eta}_T}{a_T} \leq \|\boldsymbol{N}\| - 1 + \frac{\tilde{\eta}_T}{a_T} \end{aligned}$$

Now, suppose $T \to \infty$. If $a_T \to \infty$, then $(\tilde{\eta}_T/a_T) \to 0$ because $\tilde{\eta}_T \leq d_{\max}\|\boldsymbol{N}\|$ remains finite. If a_T is bounded, then $\tilde{\eta}_T \to 0$ as the system empties, and again $(\tilde{\eta}_T/a_T) \to 0$. Therefore,

$$d_{\text{ave}} = \limsup_{T\to\infty} d_{\text{ave}}(T) \leq \|\boldsymbol{N}\| - 1.$$

Because for a given bipartite graph the cardinality of any maximal matching is at least half the cardinality of any maximum matching, the second result follows in the same manner. □

The proof of Theorem 3.1 relies on the following lemma. Recall that the row i and column j line sums of $\boldsymbol{N}(k)$ are $R_i(k)$ and $C_j(k)$, respectively.

Lemma A.2: Using any algorithm in CONTINUOUS_MAXIMAL with $\alpha \leq (1/3)$, if k is any integer such that $\|\boldsymbol{N}(k)\| \leq \alpha S$, then $\forall i,\ j \in \mathcal{N}$, at least $R_i(k)$ packets depart from row i and $C_j(k)$ packets depart from column j during slots $[k\colon k + 3\alpha S)$.

Proof: Fix k and $i \in \mathcal{N}$. The result is proved only for row i because of symmetry between rows and columns. Consider two cases. First, suppose that for each $j \in \mathcal{N}$, at least $N_{ij}(k)$ packets depart from cell (i, j) during slots $[k\colon k+3\alpha S)$. Summing over j shows that at least $R_i(k)$ packets depart from row i during slots $[k\colon k + 3\alpha S)$, and the first case is covered. In the second case, there is some $j \in \mathcal{N}$ such that less than $N_{ij}(k)$ packets depart from cell (i, j) during slots $[k\colon k + 3\alpha S)$, so that $N_{i,j}(l) \geq 1$ for $k \leq l \leq k + 3\alpha S$. Therefore, at least one packet must depart from either row i or column j during each of the $3\alpha S$ slots in $[k\colon k + 3\alpha S)$. At most $C_j(k) - N_{ij}(k) + \alpha S \leq 2\alpha S - 1$ of these $3\alpha S$ packets can be from column j, but not row i because at most αS new packets arrive at column j during slots $[k\colon k + 3\alpha S)$. Thus, at least $3\alpha S - (2\alpha S - 1) = \alpha S + 1$ packets depart row i during slots $[k\colon k + 3\alpha S)$. Because $\alpha S + 1 \geq R_i(k)$, both cases are covered, and the lemma is proved. □

Proof of Theorem 3.1: Fix any algorithm in CONTINUOUS_MAXIMAL. Argue by induction that $\|\boldsymbol{N}(k)\| \leq \alpha S,\ \forall k \in Z$. By assumption, $\|\boldsymbol{N}(k)\| = 0$ for $k < 0$. For any $k \in Z$, if $\|\boldsymbol{N}(k)\| \leq \alpha S$, then $\|\boldsymbol{N}(k + S)\| \leq \alpha S$ by Lemma A.2 and the $(\alpha,\ S)$ arrival constraint. Therefore, $\|\boldsymbol{N}(k)\| \leq \alpha S$ for all k by induction, and property (P3) holds. Now, consider the delay of any fixed packet arriving at cell $(i,\ j)$ in slot k [counted in $N_{ij}(k)$]. At most $2\alpha S$ packets arrive at row i, and at most $2\alpha S$ packets arrive at column j during slots $[k\colon k + 6\alpha S)$. Thus, there are at most $C_j(k) - 1 + R_i(k) - 1 + 4\alpha S \leq 6\alpha S - 2$ packets which can block the fixed packet during slots $[k\colon k + 6\alpha S)$. Therefore, $d_{\max} \leq 6\alpha S - 2$. By Lemma A.1, $d_{\text{ave}} \leq 2\alpha S - 1$. This completes the proof of Theorem 3.1. □

Proof of Theorem 3.2: The proof is exactly the same as that of Theorem 3.1 using Lemma A.2 with $3\alpha S$ replaced by $2\alpha S$ everywhere. To see that Lemma A.2 can be modified in the second case, note that at most $C_j(k) - N_{ij}(k) \leq \alpha S - 1$ of the $2\alpha S$ packets can be from column j, but not row i because no new packets which arrive at column j during slots $[k\colon k + 2\alpha S)$ can block packets in $N_{ij}(k)$. The maximum access delay is no more than $2(\alpha S - 1)$ because a packet in cell $(i,\ j)$ can be blocked by at most $\alpha S - 1$ row i packets and at most $\alpha S - 1$ column j packets. Clearly, $d_{\text{ave}} \leq d_{\max}$. □

Proof of Theorem 3.3: Fix $S \geq 6$. Before considering an arbitrary algorithm in the class CONTINUOUS_STATIC, a particular algorithm for an $n = 2$ station system is examined. Consider the following arrival matrix sequence during slots

$[0\colon 2l(m+1))$:

$$\overbrace{\begin{pmatrix}1&0\\1&0\end{pmatrix}, \underbrace{\begin{pmatrix}0&1\\0&0\end{pmatrix}, \begin{pmatrix}0&0\\1&0\end{pmatrix}}_{\text{Repeat } m \text{ times}}, \begin{pmatrix}0&1\\0&0\end{pmatrix}}^{\text{Repeat } l \text{ times}},$$

The inner two matrices are repeated m times, and then the whole sequence of $2m+2$ matrices is repeated l times. Consider a CONTINUOUS_STATIC algorithm with the following property: In each slot, the algorithm schedules the packet in the upper right corner cell or the lower left corner cell, ignoring the packet in the upper left corner cell. This upper left corner cell backlog can be made arbitrarily large by making l large. Because it is assumed that $\alpha > 1/2$ and αS is an integer, the sequence satisfies the (α, S) constraint for sufficiently large m. The backlog matrices in this two-station example are $\left(\begin{smallmatrix}\beta&0\\1&0\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}\beta&1\\0&0\end{smallmatrix}\right)$, where β represents any positive integer. For $n \geq 2$, any two-station pair is susceptible to the same pattern if one "corner" is ignored by the static scheduling algorithm.

Fix any algorithm X in CONTINUOUS_STATIC. Note that X must schedule at least one entry of $\left(\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right)$ or else the backlog in the two nonempty cells can be driven to infinity. If X schedules the lower left corner of the backlog matrix $\left(\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right)$, then it must schedule the upper left corner of the backlog matrix $\left(\begin{smallmatrix}1&1\\0&0\end{smallmatrix}\right)$ or an infinite cell backlog can occur as in the two-station example. Two sequences of backlog matrices are given below. Parentheses indicate the matching (containing one cell) that transmission algorithm X selects for each backlog matrix in the sequences. Once the first matching is fixed, the others are implied in order to avoid the infinite cell backlog pattern in the two-station example. Recall that the matching algorithm used by X considers only whether each element is zero or nonzero.

$$\begin{pmatrix}(1)&0&0\\1&0&0\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&0&0\\(1)&1&0\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&1&0\\0&(1)&0\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&(1)&1\\0&0&0\\0&0&0\end{pmatrix}$$

$$\begin{pmatrix}(1)&0&0\\1&0&0\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&0&0\\(1)&0&1\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&0&1\\0&0&(1)\\0&0&0\end{pmatrix} \Rightarrow \begin{pmatrix}0&1&(1)\\0&0&0\\0&0&0\end{pmatrix}.$$

There is a contradiction between the last matchings in the two sets. The theorem is proved because some two-station pair admits the infinite backlog pattern of the two-station example. □

Proof of Theorem 3.4: The proof is by induction on time k with the following proposition.

Proposition P_k: Any line backlog is zero in at least one slot during any S consecutive slots ending by time $k+S$.

Property (P4) means that P_k is true for all $k \in Z$. Clearly, P_k is true for $k < 0$ because the system starts empty. Fix $k \geq 0$. Suppose P_{k-1} is true. Fix a line, row i, without loss of generality. It is shown that $R_i(l) = 0$ for some $l \in [k\colon k+S)$. By the induction hypothesis, $R_i(l) = 0$ for some $l \in [k-1\colon k+S-1)$. If $R_i(k-1) > 0$, then $R_i(l) = 0$ for some $l \in [k\colon k+S-1)$, and therefore for some $l \in [k\colon k+S)$. On the other hand, suppose $R_i(k-1) = 0$. Let n_c be the number of different row i cells to which packets arrive during slots $[k\colon k+S)$. The columns containing these cells are called *covered columns.* Note that $n_c \leq \alpha S$, and $\alpha \leq (1/3)$ for $S \geq 1$ by the theorem assumption. Also, $\|\boldsymbol{N}(k-1)\| \leq \alpha S$ by Proposition P_{k-1} and the (α, S) constraint. Apply Lemma A.2 to see that at least $C_j(k-1)$ packets depart from any covered column j during slots $[k-1\colon k+3\alpha S-1)$. At most $n_c\alpha S$ packets arrive at covered columns (including all cells in row i) during slots $[k\colon k+S)$. Because $n_c \leq \alpha S$ and $3\alpha S + (\alpha S)^2 \leq S$ [because $\alpha \leq (\sqrt{4S+9}-3)/2S$], the key inequality $3\alpha S + n_c\alpha S \leq S$ holds. At least one packet departs from some covered column in each slot until row i is empty, so that $R_i(l) = 0$ for some $l \in [k\colon k+3\alpha S+n_c\alpha S)$. Therefore, $R_i(l) = 0$ for some $l \in [k\colon k+S)$. Thus, P_k is true, and Theorem 3.4 is proved by induction. □

Appendix B
Examples of Pathological Algorithms and Arrival Sequences

A. Example for Remark 2.1

Given $\alpha > 1/2$, $S \geq 2/(2\alpha - 1)$, and $n \geq \alpha S + 1$, an example of an $(\alpha,\ S)$ arrival sequence and an algorithm in BATCH_MAXIMUM is presented such that the algorithm does not have any of the properties (P1)–(P3). Define an $n \times n$ traffic matrix C as follows:

$$C = \begin{cases} \alpha S - 1, & \text{if } i = j \text{ and } i \leq \alpha S \\ 1, & \text{if } j = \alpha S + 1 \text{ and } i \leq \alpha S \\ 0, & \text{otherwise.}\end{cases}$$

It is possible to decompose C into $2\alpha S - 1$ switching matrices such that each matrix in the sequence is a maximum matching for the traffic left after the previous matrices in the sequence have been subtracted from C. For brevity, we simply illustrate this decomposition for the case $\alpha S = 3$ and $n = 4$:

$$\begin{pmatrix}2&0&0&1\\0&2&0&1\\0&0&2&1\\0&0&0&0\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{pmatrix}.$$

Assume for the rest of this example that the arrival sequence is the following $(\alpha,\ S)$ traffic sequence: once every S slots the arrival matrix is C, and no arrivals occur in other slots.

We next select an algorithm in BATCH_MAXIMUM , first assuming that the duration L of the batch intervals is given by $L = S$. The new traffic added at the beginning of each batch interval has traffic matrix C. Assume that the algorithm attempts to schedule the traffic as in the above decomposition of C. Since $S < 2\alpha S - 1$, the algorithm will not finish scheduling the first batch before the second batch is to be considered. The algorithm can continue scheduling packets from each batch in an identical way, and the backlog tends to infinity. Therefore, none of the properties (P1)–(P3) holds.

We note at this point that an algorithm in CONTINUOUS_MAXIMUM can be constructed to behave in the same way, so that properties (P1)–(P3) do not hold for some algorithms in CONTINUOUS_MAXIMUM (or CONTINUOUS_MAXIMAL) as well. We now return to batch algorithms.

If, instead, the batch interval is to be taken to be $L = kS$ for some $k \geq 2$, then the same arrival process leads to new traffic at the beginning of each batch given by kC. By repeating each matrix on the right side k times in the above decomposition of C, we obtain a similar decomposition of kC into $k(2\alpha S - 1)$ switching matrices, which again exceeds the batch scheduling interval $L = kS$. Thus, the backlog tends to infinity, and none of the properties (P1)–(P3) holds. Finally, a similar overflow also occurs if the batch interval is not a multiple of L.

B. Example for Remark 3.1

An arrival sequence and a scheduling algorithm in CONTINUOUS_MAXIMUM are presented such that $R_1(k) > 0,\ \forall k \in Z$; that is, the row 1 line sum is always nonzero after time 0. Suppose $\alpha,\ S$, and n are given and $n \geq \alpha S$. There are arrivals only in slots $b\alpha S$ for $b = 0,\ 1,\ 2,\ \cdots$, described as

$$A_{ij}(b\alpha S) = \begin{cases} 1, & \text{if } j = (b \bmod \alpha S) + 1 \text{ and } i \leq \alpha S \\ 0, & \text{otherwise} \end{cases}$$

where mod is the integer modulo operator. This arrival sequence repeats every $(\alpha S)^2$ slots. For this sequence to be $(\alpha,\ S)$, the condition $(\alpha S)^2 \geq S$ is required. This is equivalent to $\alpha \geq (1/\sqrt{S})$. In slot k, the matching algorithm schedules a single departure from cell $(i,\ j)$ where $i = \alpha S - (k \bmod \alpha S)$ and $j = (\lfloor k/\alpha S \rfloor \bmod \alpha S) + 1$. In any slot, only one column is nonempty, but row 1 is never empty at a backlog measurement time. Therefore, (P4) does not hold. In summary, if $\alpha,\ S$, and n are such that αS is an integer with $n \geq \alpha S$ and $\alpha \geq (1/\sqrt{S})$, then there is an $(\alpha,\ S)$ arrival sequence and a CONTINUOUS_MAXIMUM algorithm such that one line sum is always nonzero.

References

[1] R. Ramaswami, “Multiwavelength lightwave networks for computer communication,” *IEEE Commun. Mag.*, vol. 31, pp. 78–88, Feb. 1993.

[2] M. Chen, N. Dono, and R. Ramaswami, “A media-access protocol for packet-switched wavelength division multiaccess metropolitan area networks,” *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1048–1057, Aug. 1991.

[3] N. Dono, P. Green, K. Liu, R. Ramaswami, and F. Tong, “A wavelength division multiple access network for computer communication,” *IEEE J. Select. Areas Commun.*, vol. 8, pp. 983–994, Aug. 1991.

[4] T. K. Chang *et al.*, “Implementation of STARNET: A WDM computer communications network,” *IEEE J. Select. Areas Commun./J. Lightwave Technol.* (Special Issue on Optical Networks), vol. 14, pp. 824–839, June 1996.

[5] B. Hajek and T. Weller, “Scheduling nonuniform traffic in a packet-switching system with large propagation delay,” *IEEE Trans. Inform. Theory*, vol. 41, pp. 358–365, Mar. 1995.

[6] M. Chen and T.-S. Yum, “A conflict free protocol for optical WDMA networks,” in *Proc. IEEE GLOBECOM*, Dec. 1991, pp. 1276–1281.

[7] L. Tassiulas, “Scheduling and performance limits of networks with constantly changing topology,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 1067–1073, May 1997.

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

[9] R. Cruz, “A calculus for network delay, Part I: Network elements in isolation,” *IEEE Trans. Inform. Theory*, vol. 37, pp. 114–131, Jan. 1991.

[10] A. Descloux, “Stochastic models for ATM switching networks,” *IEEE J. Select. Areas Commun.*, vol. 9, pp. 450–457, Apr. 1991.

[11] J. Kurose, “On computing per-session performance bounds in high-speed multi-hop computer networks,” in *Proc. ACM SIGMETRICS 1992*, New Port, RI, June 1992, pp. 128–139.

[12] L. G. Valiant and G. J. Brebner, “Universal schemes for parallel communication,” in *Proc. 13th Annu. ACM Symp. Theory of Computing*, May 1981, pp. 263–277.

[13] T. Inukai, “An efficient SS/TDMA time slot assignment algorithm,” *IEEE Trans. Commun.*, vol. 27, pp. 1449–1455, Oct. 1979.

[14] I. Gopal and C. Wong, “Minimizing the number of switchings in an SS/TDMA system,” *IEEE Trans. Commun.*, vol. COM-33, pp. 497–501, June 1985.

[15] M. Hall, Jr., *Combinatorial Theory*. Waltham, MA: Blaisdell, 1969.

[16] C. Papadimitriou and K. Steiglitz, *Computational Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[17] H. Gabow and O. Kariv, “Algorithms for edge coloring bipartite graphs and multigraphs,” *SIAM J. Comput.*, vol. 11, pp. 117–129, Feb. 1982.

[18] K. Eng, M. Karol, and Y. Yeh, “A growable packet (ATM) switch architecture: Design principles and applications,” *IEEE Trans. Commun.*, vol. 40, pp. 423–430, Feb. 1992.

[19] G. Bongiovanni, D. Coppersmith, and C. Wong, “An optimum time slot assignment algorithm for an SS/TDMA system with variable number of transponders,” *IEEE Trans. Commun.*, vol. COM-29, pp. 721–726, May 1981.

[20] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, “Input versus output queueing on a space-division packet switch,” *IEEE Trans. Commun.*, vol. COM-35, pp. 1347–1356, Dec. 1987.

[21] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[22] L. Zhang, “A new architecture for packet-switching network protocols,” Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, 1989.

[23] C. Liu, *Topics in Combinatorial Mathematics.* Mathematical Association of America, 1972.

Timothy Weller received the B.S. and M.S. degrees in electrical engineering from Michigan State University in 1987 and 1988, respectively, and the Ph.D. in electrical and computer engineering at the University of Illinois at Urbana–Champaign in 1993.

Since July 1993, he has been with Donaldson, Lufkin & Jenrette, New York, where he is currently Vice President, Research Department. As a graduate student, he was a National Science Foundation Graduate Fellow, an AT&T Bell Laboratories Scholar, and IEEE Frank Cowan Communications Scholar. Since 1980, he has held industry positions as a video game designer, a computer science instructor, a VLSI software engineer, a computer learning tools developer, and a business software applications consultant.

Bruce Hajek (M’79–SM’84–F’89) received the B.S. degree in mathematics and the M.S. degree in electrical engineering from the University of Illinois in 1976 and 1977, respectively, and the Ph.D. degree in electrical engineering from the University of California at Berkeley in 1979. He is a Professor in the Department of Electrical and Computer Engineering and in the Coordinated Science Laboratory at the University of Illinois at Urbana–Champaign, where he has been since 1979. He served as Associate Editor for Communication Networks and Computer Networks for the IEEE Transactions on Information Theory (1985–1988), as Editor-in-Chief of the same Transactions (1989–1992), and as President of the IEEE Information Theory Society (1995). His research interests include both wireless and highspeed communication networks, stochastic systems, combinatorial and nonlinear optimization, and information theory.