

Random Algorithms for Scheduling Multicast Traffic in WDM Broadcast-and-Select Networks

Eytan Modiano, *Member, IEEE*

Abstract— We develop and analyze simple algorithms for scheduling multicast traffic in wavelength division multiplexing (WDM) broadcast-and-select networks with N nodes, W wavelengths, and a single receiver per node that can be tuned to any of the W wavelengths. Each message is addressed to k randomly chosen nodes. Since optimal message scheduling in a WDM network is known to be very difficult, we study two simple scheduling schemes: in the first, a message is continuously retransmitted until it is received by all of its intended recipients; and in the second, a random delay is introduced between retransmissions of the same message. We develop a throughput analysis for both schemes using methods from discrete-time queueing systems and show that the algorithm with random delays between retransmissions results in higher throughput. We also consider a number of receiver algorithms for selecting among multiple simultaneous transmissions and show, through simulation, that an algorithm where the receiver selects the message with the least number of intended recipients performs better than a random selection algorithm. Finally, we show that channel utilization can be significantly increased with multiple receivers/node.

Index Terms— Broadcast star topology, lightwave networks, local lightwave networks, multicast/broadcast algorithms, multicast scheduling algorithms, multicast switching, wavelength division multiplexing.

I. INTRODUCTION

THIS paper examines the problem of scheduling multicast transmissions in wavelength division multiplexing (WDM) local area networks. Recently, multicasting has received a great deal of attention in the context of higher layer protocols for wide area networks [1]–[2]. Increasingly applications and protocols come to depend on the underlying networks for providing a multicast capability. The emergence of WDM LAN technology [4] gives rise to the multicasting problem in this environment. In order for WDM LAN's to gain widespread use they must support a multicasting capability.

Most LAN's use a broadcast medium which makes multicasting simple and efficient. Every transmission is received by all of the nodes in the network. In WDM broadcast LAN's, such as a broadcast star, the problem is significantly more complicated. In WDM networks, the fiber supports multiple channels, each corresponding to a different wavelength. In

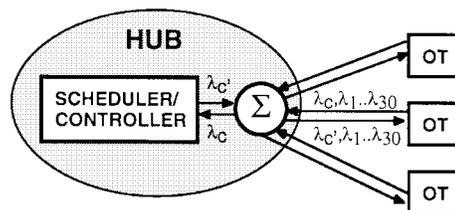


Fig. 1. A WDM-based LAN using a broadcast star and a scheduler.

order to receive a transmission on one of the channels, nodes must tune their receiver to the appropriate wavelength. If each node has only a single receiver, a transmission on one channel may not be received by nodes that are tuned to receive a message on another wavelength. In the absence of a scheduling algorithm to coordinate the transmissions, multiple simultaneous transmissions to the same receiver are possible, resulting in a reduction in the system throughput. This problem is similar to that of scheduling traffic in an input queued switch, for which it was shown in [13], [17] that for unicast traffic, under uniform traffic conditions, the throughput of an $N \times N$ switch is limited to 58%.

This paper is motivated by the design of a very high-speed WDM LAN extension to a wideband all-optical WDM network [3]. This LAN will consist of about 100 nodes connected in a broadcast star topology using 32 wavelengths (channels). Each channel will operate at a transmission rate of about 10 Gb/s. Each node will have one fast tunable transmitter and one fast tunable receiver. A node that wishes to send a message chooses a channel on which to transmit and notifies the intended receivers of that channel so that they can tune to it. The network will use a MAC protocol that allows the nodes to efficiently share the WDM channels. There are a number of such protocols that can achieve high utilization [4]–[7].

The proposed system is based on the protocol in [4], which uses a simple master/slave scheduler, as shown in Fig. 1, that is able to schedule transmissions efficiently and overcome the effects of propagation delays and transceiver tuning delays. All nodes send their requests to the scheduler on a dedicated control wavelength λ_C . The scheduler, located at the star, schedules the requests and informs the nodes on a separate wavelength λ_C' of their turn to transmit. Upon receiving their assignments, nodes immediately tune to their assigned wavelength and transmit. Similarly, the scheduler informs nodes of when and on which wavelengths they should be receiving transmissions. While the use of a centralized scheduler causes some additional delays (for sending requests and receiving scheduling assignment), it allows for very efficient scheduling

Manuscript received February 2, 1998; revised December 24, 1998 and March 29, 1999; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor B. Mukherjee. This work was supported by the Defense Advanced Research Project Agency under the Next Generation Internet initiative.

The author was with the MIT Lincoln Laboratory, Lexington, MA 02420 USA. He is now with the Laboratory for Information and Decision Systems, MIT, Cambridge, MA 02139 USA (e-mail: modiano@mit.edu).

Publisher Item Identifier S 1063-6692(99)04944-4.

of transmissions. Furthermore, by measuring the amount of time that nodes take to respond to the assignments, the scheduler is able to obtain an estimate of each node's round-trip delay to the hub. This estimate is used by the scheduler to overcome the effects of propagation delays. While the details of the system are beyond the scope of this paper (e.g., the control channel access protocol, the scheduling of unicast traffic, and the measurement of propagation delays), here we focus on the problem of scheduling multicast transmissions over this system.

There are a number of previous works that address the multicast problem in a similar context [8]–[12]. In [9], a reservation-based protocol that schedules multicasts to avoid any receiver conflicts is described. Since the protocol of [9] requires all of the intended receivers to be available before a message can be transmitted, it results in low receiver utilization (e.g., receivers spending much of their time without being able to receive anything). This is a problem, in particular, when the size of the multicast group is large and many receivers must be free at once. If receivers were not forced to wait for one another to become available, much higher utilization can be achieved. This is especially true when the system is heavily loaded (e.g., large multicast group) and all receivers have some message which they can be receiving. Consequently, in [10], an alternative reservations protocol is described that schedules multicasts to take place in multiple transmissions, thereby reducing the amount of time that available receivers have to wait before receiving the message. Finally, in [11], a scheduling algorithm is described for performing multicasts in an $N \times N$ switch. Since this problem is similar in nature to the one we consider here, this algorithm may also be applicable to WDM broadcast LAN's.

While the above algorithms attempt to increase system utilization through the design of an efficient scheduler, they require the implementation of sophisticated scheduling algorithms. In a network operating at 10 Gb/s per WDM channel, millions of such schedules may have to be generated every second. This enormous throughput requirement significantly limits the sophistication of the scheduling algorithm that can be employed. We are therefore driven to consider simpler algorithms based on random scheduling. Aspects of random scheduling of multicast traffic have been considered in the past in the context of an $N \times N$ switch. In [12], an approximate analysis is given for a random packet selection policy in an $N \times N$ switch, with mixed input traffic, under the assumption that all copies of a packet have to be transmitted during the same slot.

In this paper, we examine the performance of random multicast transmissions in a WDM network. We compare two random scheduling schemes; in one, a message is continuously retransmitted until it is received by all of its intended recipients, and in the other, a random delay is introduced between retransmissions of the same message. This random delay is introduced in order to alleviate the effects of head-of-line (HOL) blocking. We show that the algorithm with random delays between retransmission results in higher throughput. We also study a number of receiver algorithms for selecting among multiple simultaneous transmissions and show that an

algorithm where the receiver selects the message with the least number of intended recipients performs better than a random selection algorithm. Finally, we study the impact of having multiple receivers/node and show that channel utilization can be significantly improved by having multiple receivers/node. Hence, the results of this paper have both practical value for the implementation of simple multicast scheduling algorithms (MSA's) in very high-speed WDM LAN's, and theoretical value for providing insight into the performance of multicast switching systems.

We assume first-come-first-serve (FCFS) scheduling of traffic at the different nodes and a random selection policy for choosing which nodes transmit. We assume a slotted system, where the size of a slot is equal to the amount of time that it takes to transmit a message. This time also includes the transceiver tuning delays. When tuning delays are small compared to message transmit times, these tuning delays are only a fraction of a slot time. The system proposed here will use very fast transceivers that can tune in just a few nanoseconds. Such transceivers are becoming commercially available and will be used to build the system proposed in [4]. The network consists of N nodes and W wavelengths, and each message is addressed (multicast) to k receivers (i.e., destination set size k is fixed) randomly chosen from the N nodes. During each slot, W of the N nodes are chosen to transmit, and each receiver tunes to one of the wavelength that has a message addressed to it. Naturally, only those nodes that actually have a message to send are chosen to transmit. Messages are transmitted repeatedly until they are received by all of their intended receivers.

Notice, that in the presence of a scheduler, all decisions regarding which nodes transmit and which transmissions nodes choose to receive can be made by the scheduler. Hence, the scheduler will always know the state of the system instantaneously (e.g., which messages need to be retransmitted to which nodes, etc.), and will be able to schedule retransmissions immediately. Hence, no additional protocol overhead and delays are needed in order to coordinate retransmissions of a message.¹

For the purpose of analyzing throughput, we assume that the system is constantly backlogged. That is, all of the nodes always have a message to transmit. In this context, the system throughput can be expressed as the average number of multicast message transmissions completed per slot per wavelength (channel). When the system is backlogged, it is the inverse of the average number of transmissions required per successful multicast. A lower bound on this number can be easily obtained by observing that, even with an optimal scheduling algorithm, during every slot each node can at most receive one message transmission.

The system consists of W channels, simultaneously transmitting multicast messages each intended for k randomly chosen nodes. Since there are N nodes, on average, each node has kW/N transmissions intended for it. Since each node can only receive one transmission at a time \bar{T} , the average number

¹Of course, the system will always incur the overhead of transmitting the scheduling assignments over the dedicated control channel.

of transmissions required per message, is lower bounded by

$$\bar{T} > \max\left(\frac{kW}{N}, 1\right). \quad (1)$$

Therefore, regardless of whether or not a scheduling algorithm is used, or which algorithm is used, the system throughput, expressed in terms of completed multicasts transmissions per slot per wavelength, cannot exceed $1/\bar{T}$. It is interesting to note that when kW is less than N , the system is channel limited; that is, there are not enough channels to keep all of the receivers busy. In the channel-limited case, receivers can never be fully utilized. When kW is greater than N , the system is receiver limited; that is, the number of receivers is too small to keep all of the channels busy with new transmissions. In the receiver-limited case, channels can never be fully utilized because messages will have to be transmitted multiple times, and the lower bound gives the average number of required transmissions when receivers are fully utilized. The throughput efficiency of a multicast algorithm can therefore be defined as the ratio of the lower bound from (1) to the average number of transmissions that the algorithm requires per successful multicast. When the system is channel limited (lower bound = 1), this simply measures the throughput of the channels (the average number of completed multicast transmissions per slot/wavelength). However, when the system is receiver limited this measure compares the channel throughput to the maximum achievable throughput and, in fact, measures the receiver utilization. We believe that this measure gives a better indication of the efficiency of a scheduling algorithm.

In this paper, we consider simple MSA's and compare their performance to this lower bound. We start, in Section II, by considering a protocol where a message is repeatedly transmitted until it is received by all of its intended recipients. This protocol leads to a form of HOL blocking, where a message waiting on a busy receiver blocks the channel from being used by other messages that are addressed to unoccupied receivers. To remedy this problem, in Section III, we consider a protocol which retransmits a message only after waiting a random amount of time to allow busy receivers to become unoccupied. We show that this protocol offers a throughput improvement over the one in Section II. In Section IV, we show that an additional throughput improvement can be achieved if receivers intelligently select the transmission to which they listen and finally, in Section V, we study the impact of having multiple receivers/node.

II. RANDOM SELECTION WITH PERSISTENT RETRANSMISSION

With this protocol, whenever a channel becomes available, a message is chosen for transmission, at random, from among the available nodes that have a new message to send. The message is repeatedly transmitted until it has been received by all of its intended recipients (all members of the multicast group). In this protocol, no new transmission can begin on a channel until the previous one is completed. Hence, a message occupies the channel continuously until it is received by all of its intended recipients. When the channel becomes available, a

new message is chosen, at random, from among the free nodes (not otherwise busy with a previous multicast transmission). If during the m th time slot, C_m multicast transmissions were completed, then up to C_m new transmissions can begin in the $(m+1)$ th slot. Those transmissions will be chosen at random from amongst the free nodes.

In order to analyze the achievable throughput of this protocol, we assume that the system is constantly backlogged. That is, as soon as the transmission of one message is completed, a new message is immediately available for transmission.² Each message is destined to exactly k receivers chosen at random from the N nodes (i.e., destination set size is k). Hence, each packet has a probability k/N of being addressed to any given node. When a node has more than one packet addressed to it, it must choose only one of them to receive. The choice of which packet a node should choose to receive is addressed in more detail in Section IV. Here, we make the assumption that the receiver selects the packet based on the time it was first transmitted in a FCFS order. If two or more packets were first transmitted in the same slot, then the receiver chooses amongst them at random (in practice, this random selection of which message to receive can be made by the scheduler which informs the nodes of the decision). Here, we use this FCFS ordering mainly because it simplifies the analysis. However, in Section IV we will show, through simulation, that FCFS ordering results in shorter delays and increased throughput over a random selection policy.

The analysis that follows assumes the use of a centralized scheduler, as shown in Fig. 1. This scheduler can, in effect, make all transmission and reception decisions and inform the appropriate nodes when to transmit and which message to receive. As a result, the scheduler has instant feedback regarding the state of the system and can schedule retransmissions immediately, without incurring any additional delays or overheads for coordination of retransmissions.

Consider the m th time slot, and let Q_m^i be the number of messages addressed to node i at the end of the m th time slot. Since during any time slot, node i can only receive one message, it follows that

$$Q_m^i = \max(0, Q_{m-1}^i + A_m^i - 1) \quad (2)$$

where A_m^i is the number of new messages arriving at the start of the m th time slot and destined to node i .³ A new message arrives at the beginning of a time slot only when the transmission of a previous message has been completed at the end of the previous slot, and that new message is destined to node i with probability k/N . Hence, if we let C_{m-1} denote the number of completed multicast transmissions during the $(m-1)$ th time slot, then C_{m-1} also denotes the number of newly arrived messages at the beginning of the m th time slot. Now A_m^i has the binomial distribution

$$\Pr[A_m^i = l] = \binom{C_{m-1}}{l} \left(\frac{k}{N}\right)^l \left(1 - \frac{k}{N}\right)^{C_{m-1}-l} \quad (3)$$

²Here we are not concerned with the channel assignment problem and assume that a protocol, such as the one described in [4], is used that can coordinate the transmissions so that the channels are fully utilized.

³We assume that packets arriving at the start of a slot can be transmitted during that slot.

and \bar{A} , the average number of new arrivals at a node (per slot) is equal to $\bar{C}k/N$, where \bar{C} is the average number of completed multicast transmissions per slot. It can be shown [13] that, for finite values of k , as W and N (the number of channels and nodes) approach infinity, A_m^i becomes Poisson of rate \bar{A} . Also, noticing that (2) is the same as that of an M/D/1 queue [14], we can express \bar{Q} , the average number of messages destined to a receiver, according to the well-known M/D/1 formula [14]

$$\bar{Q} = \bar{A} + \frac{(\bar{A})^2}{2(1-\bar{A})}.$$

Now we turn our attention to the computation of \bar{C} , the average number of completed multicast transmissions per slot. We arrive at this number by first computing the average number of transmissions (slots) required per successful multicast message. For a multicast message to be successful, it must be received by all k nodes for which it was intended. Consider a message destined to nodes n_1, \dots, n_k and suppose that when the message arrives it finds N_1, \dots, N_k other messages waiting to be transmitted to nodes n_1, \dots, n_k , respectively. These values represent the number of messages waiting to be received by nodes n_1, \dots, n_k . They can be viewed as implicit output queues at these nodes. Of course, actual output queues do not exist at the nodes, but these "implicit" queues can be used for the analysis. Because of the FCFS receiver service discipline, the transmission of the message will not be completed until it is received by the node with the largest queue.

Let $N_i = q_i + r_i$, where q_i is the number of messages waiting in the queue, and r_i is the number of messages at the transmitter (r_i is the equivalent of the residual time for the message at the head of the queue in an unslotted system). That is, $r_i = 1$ if $N_i > 0$ and 0 otherwise. If we now let N_{\max} be the number of messages waiting at the node with the busiest queue (waiting in queue and at the transmitter), and let Q_{\max} be the number of messages waiting at the busiest queue, then $N_{\max} = Q_{\max} + R_{\max}$ and $R_{\max} = N_{\max} - Q_{\max}$. Notice that the average value of R_{\max} is also equal to the average number of new arrivals to the busiest node at the beginning of the slot. Due to the nature of the slotted system where all new messages arrive at once as a batch, on average, a message has to wait only for half of the new arrivals because only half of the new arrivals will be placed ahead of it in the queue. That is, if there are new arrivals to the queue during a slot, on average, only half of those new arrivals will be placed ahead of the given message in the queue and half would be placed behind it. Hence, if \bar{R}_{\max} is the average number of new arrivals to the queue during a slot any message arriving will have to wait for only half of \bar{R}_{\max} . Accounting for the additional transmission time of one slot the average wait in the largest queue would then be

$$\bar{T} = 1 + \bar{Q}_{\max} + \frac{\bar{R}_{\max}}{2}. \quad (4)$$

In order to compute \bar{T} , we must first compute the average value of $Q_{\max} = \max(q_1, \dots, q_k)$. It can be shown that for k finite, as N approaches infinity, $q_1 \dots q_k$ behave as k

independent M/D/1 queues. Hence, we wish to compute the average value of the maximum queue size of k independent M/D/1 queues.

Let $Q_{\max} = \max(q_1, \dots, q_k)$. Then

$$\begin{aligned} \Pr(Q_{\max} \leq l) &= \Pr(q_1 \leq l) \Pr(q_2 \leq l) \dots \Pr(q_k \leq l) \\ &= \Pr(q_1 \leq l)^k \end{aligned} \quad (5)$$

and

$$\begin{aligned} \Pr(Q_{\max} = l) &= \Pr(Q_{\max} \leq l) - \Pr(Q_{\max} \leq l-1) \\ &= \Pr(q_1 \leq l)^k - \Pr(q_1 \leq l-1)^k. \end{aligned} \quad (6)$$

Thus

$$\begin{aligned} \bar{Q}_{\max} &= \sum_{l=1}^{l=\infty} l \times \Pr(Q_{\max} = l) \\ \bar{Q}_{\max} &= \sum_{l=1}^{l=\infty} l \times [\Pr(q_1 \leq l)^k - \Pr(q_1 \leq l-1)^k]. \end{aligned} \quad (7)$$

Lastly, \bar{R}_{\max} (the average number of messages at the head of the queue) is simply equal to the probability that the busiest queue is not empty.⁴

Since the busiest queue is empty only if all of the queues are empty, we have

$$\bar{R}_{\max} = 1 - (1 - \bar{A})^k \quad (8)$$

and

$$\begin{aligned} \bar{T} &= 1 + \frac{1 - (1 - \bar{A})^k}{2} + \sum_{l=1}^{l=\infty} l \times [\Pr(q_1 \leq l)^k \\ &\quad - \Pr(q_1 \leq l-1)^k]. \end{aligned} \quad (9)$$

Notice that in order for the system to be stable, the queues at the receivers [governed by (2)] must be finite. Since (2) represents an M/D/1 queue, the arrival rate \bar{A} must be less than 1. If \bar{A} is greater than 1, then \bar{T} will be infinite and throughput will be zero. Hence, in obtaining the maximum throughput, by definition, we are assured that \bar{A} will be less than 1. So far, we have expressed \bar{T} in terms of the probability distribution of the number of messages in a single M/D/1 queue. This distribution can be obtained by solving the corresponding Markov chain [14]. While we do not know of closed-form results for this distribution, numerical evaluation is rather straightforward. Since the transmission takes place over W channels in parallel $\bar{C} = W/\bar{T}$. Finally, we obtain an expression for \bar{A}

$$\bar{A} = \left(\frac{W}{\bar{T}}\right) \left(\frac{k}{N}\right). \quad (10)$$

Notice again that the above expression is only exact when W and N are infinite. For finite values of W and N , this analysis is only approximate, but, as will be shown shortly, it performs very well when compared to simulations. Equation (10) gives the arrival rate of new messages to each of the independent M/D/1 queues representing the number of

⁴Notice that while the system is constantly backlogged, an output queue can still be empty if none of the transmissions on the w wavelengths are intended for it.

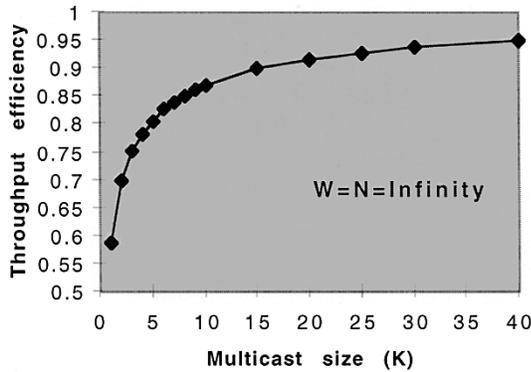


Fig. 2. Throughput efficiency versus multicast group size.

messages waiting to be received by a given node. This equation is expressed in terms of \bar{T} , which is given by (9). However, (9) is expressed in terms of the statistics of an M/D/1 queue with arrival rate \bar{A} . Hence, the two equations must be solved together in order to obtain the values of \bar{A} and \bar{T} . This can be done using an iterative algorithm similar to the one described in [15], [16].

In Fig. 2, we plot the throughput efficiency versus k when $W = N = \text{Infinity}$. Notice that in this case, the throughput efficiency is simply equal to $\bar{A} = k/\bar{T}$. This is the special case where the number of channels is equal to the number of nodes. This case has received much attention in the literature because it also applies to an $N \times N$ switch. As can be seen from the figure, with $k = 1$, the maximum throughput of 0.586 can be achieved. This is the well-known result of [13]. It is interesting to note, however, that as we increase k , the throughput increases. This increase in channel utilization is due to an increase load on the receivers. As k increases, receivers become heavily utilized, but at the same time the probability of receiver conflicts is increased. However, the significance of this result is that with large k there is very little to be gained by trying to efficiently schedule multicast traffic. This is a surprising result that leads us to think that random scheduling of multicast traffic can achieve good throughput efficiency.

The results shown in Fig. 2 are obtained by solving (9) and (10), which are only exact for infinite values of N and W . With finite N and W , the analysis leading to (9) and (10) is approximate for two reasons: first, with finite W , the arrivals at the different “implicit” output queues would no longer be Poisson; and second, with finite N , the output queues would not be independent. We rely on both of these assumptions to arrive at our results. However, it is interesting to note that this analysis provides a rather accurate approximation for finite values of N and W . In Table I, we compare the exact results for $W = N = \text{Infinity}$ with those arrived at via simulation for finite values of N and W . As can be seen from the table, with $W = N = 100$, the exact results (for $W = N = \text{infinity}$) are well within 1% of the simulation results. Hence, it appears that the exact results for a system with an infinite number of nodes and channels provide a rather good approximation for a system with a finite (even small) number of nodes and wavelengths.

In Fig. 3, we consider what may be a more relevant case for a WDM network. In this figure, we plot the throughput

TABLE I
COMPARISON OF THE EXACT ANALYSIS (INFINITE NODES) TO SIMULATION WITH FINITE NUMBER OF NODES AND WAVELENGTHS

k	Exact	Simulation	Simulation
	W=N=Infinitt	W=N=32	W=N=100
	γ		
1	0.586	0.593	0.588
2	0.700	0.704	0.702
3	0.750	0.756	0.752
4	0.784	0.789	0.786
5	0.808	0.813	0.810
6	0.826	0.831	0.828
7	0.841	0.846	0.843
8	0.853	0.857	0.854
9	0.863	0.867	0.864
10	0.871	0.875	0.872

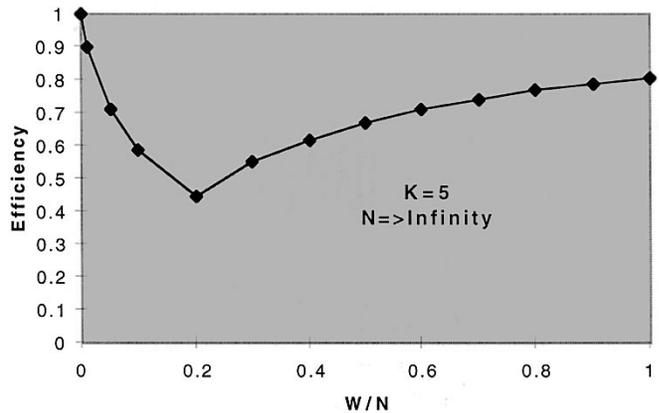


Fig. 3. Throughput efficiency versus W/N .

efficiency versus W/N for $k = 5$. Here, we define the throughput efficiency as the ratio of the lower bound of (1) to \bar{T} , the average number of transmissions per successful multicast. What we see from Fig. 3 is that when the number of channels is very small the efficiency approaches 1. This is because when the number of channels is small, the likelihood of a receiver collision (contention) is low, and hence, the efficiency is high. In contrast, when the number of channels is very high, we see that the throughput approaches the value of 0.808, which is the throughput with $k = 5$ and $N = W = \text{Infinity}$ (see Fig. 2 or Table I).

We should point out, however, that this apparent improvement in efficiency is somewhat artificial because it is due to an increased load on the receivers. The interesting region is when the receivers are neither lightly nor heavily loaded. It is interesting to note that the efficiency is minimized when $W/N = 0.2$. This represents a receiver load of 1 packet per slot per receiver (or 100%).

III. RANDOM SELECTION WITH BACKOFF RETRANSMISSION

The throughput of the persistent retransmission protocol is limited by a form of HOL blocking which results from

a channel being inefficiently used while the message being transmitted on that channel is waiting for receivers to become available. It is known that for unicast traffic, the effect of HOL blocking is partly due to the high correlation among HOL messages from slot to slot [13]. That is, if the HOL message is blocked during a given slot, the probability that it will be blocked again during the next slot is increased. In [13], it was shown that for unicast traffic, throughput can be improved if during every slot, the HOL destinations are new. This was accomplished in [13] by assuming that all HOL messages that failed to be received during a slot due to receiver contention are dropped. Of course, in a practical system, this cannot be done. However, here we explore an alternative approach to achieve a similar effect of reducing the slot-to-slot correlation. Instead of immediately retransmitting messages that were received by all of their intended recipients, we introduce a random delay between retransmissions and allow new messages to be transmitted while the old messages wait for their turn to be retransmitted. This mechanism, in effect, reduces the correlation among HOL messages from slot to slot, thereby alleviating the HOL blocking problem. Of course, this comes at the expense of the additional delay associated with a backoff algorithm.

Here, we are not concerned with the protocol involved in the coordination of these transmissions, nor with the details of the backoff algorithm.

For the analysis, we assume that this random delay between retransmission removes the correlation among the HOL messages and that the destinations of the different HOL messages chosen for transmission during a given slot are independent of one another. We also assume that when a node has multiple transmissions to choose from, it selects one at random. The issue of received selection policy will be addressed in more detail in the next section. Again, we are interested in computing the average number of times a message has to be transmitted in order for it to be received by all of its intended recipients.

During each slot, W messages are chosen, at random, for transmission from among the N nodes. These messages can be either new multicast transmissions with K intended recipients (destinations) or they can be retransmissions of previously transmitted messages, in which case they have fewer than K intended recipients. Let X be the average number of intended recipients for a multicast transmission during a slot. The first time a message is transmitted $X = k$. In subsequent transmissions, the number of intended recipients decreases, as some recipients have already received the message previously. Clearly $1 \leq X \leq k$.

Let P be the probability that an arbitrary node is addressed by an arbitrary multicast transmission (occurring on one of the W channels) during a slot. Clearly

$$P = \frac{X}{N}. \quad (11)$$

Let N_j be the number of transmissions intended for node j . Then, the probability distribution of N_j is given by

$$P(N_j = l) = \binom{W}{l} P^l (1 - P)^{W-l}. \quad (12)$$

Suppose now that node j is addressed by a multicast transmission, the probability that node j will choose to receive this particular transmission is equal to the probability that node j selects this transmission out of all of the transmissions addressed to it during this slot. Notice that this is not the same as the probability that a transmission is selected given that at least one transmission to node j has occurred. Rather, this probability is conditional on the fact that a particular transmission is addressed to node j . Therefore, if we know that a transmission is addressed to node j , the probability that it is selected depends on how many of the remaining $W - 1$ transmissions will also address node j . This probability is given by

$$P_s = \sum_{l=0}^{W-1} \binom{1}{l+1} \binom{W-1}{l} P^l (1 - P)^{W-l-1} \quad (13)$$

and $P_{ns} = 1 - P_s$ is the probability that the node does not select this particular transmission. Next we must compute X , the average number of nodes addressed by a multicast transmission. This number represents the average number of intended recipients on each wavelength.

Let X^i be the number of nodes addressed during the i th transmission attempt of a message. Then $X^1 = k$ and X^2 is equal to the number of nodes that failed to receive the transmission in the first attempt. In general, X^r is equal to the number of nodes that have not received the message by the r th attempt.

The probability that a node requires r or more transmissions is equal to the probability that a node failed to receive the message during the first $r - 1$ transmissions and is equal to $(1 - P_s)^{r-1} = (P_{ns})^{r-1}$. Since each message has k destinations, the probability that X^r is equal to l is the probability that exactly l of the k nodes require r or more transmissions. Hence

$$P(X^r = l) = \binom{k}{l} [(P_{ns})^{(r-1)}]^l [1 - (P_{ns})^{(r-1)}]^{k-l}. \quad (14)$$

The probability that a message is transmitted exactly r times is equal to the probability that r or more transmissions are required minus the probability that $r+1$ or more transmissions are required. Now, the probability that at least r transmissions are required is equal to the probability that at least one node requires r or more transmissions and is given by

$$P_t(r \leq \#trans) = 1 - (1 - (P_{ns})^{r-1})^k. \quad (15)$$

Finally, the probability that exactly r transmissions are needed is given by

$$P_t(r) = (1 - (P_{ns})^r)^k - (1 - (P_{ns})^{r-1})^k. \quad (16)$$

Therefore, the average number of transmissions required is

$$E[\text{Trans}] = \sum_{r=1}^{\infty} r [(1 - (P_{ns})^r)^k - (1 - (P_{ns})^{r-1})^k]. \quad (17)$$

So far, we have obtained the average number of transmission attempts per message. In order to compute X , the average number of intended recipients per attempt, we must first obtain NR , the average number of intended recipients for a message

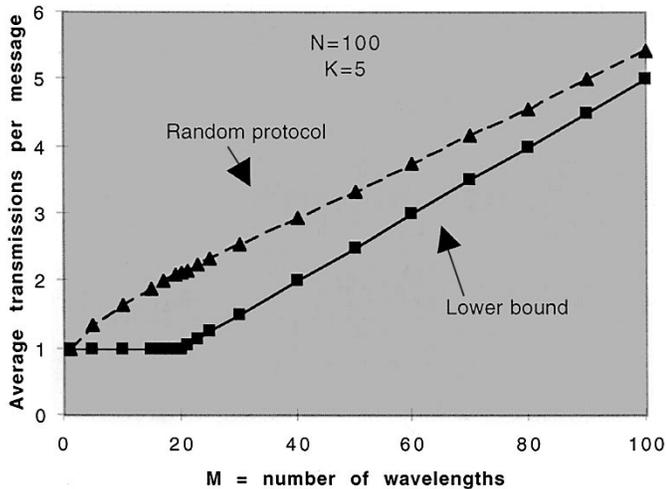


Fig. 4. Average number of transmissions per multicast message.

over all transmission attempts. Recalling that X^r is equal to the number of intended recipients in the r th transmission of the message, we have

$$NR = \sum_{r=1}^{\infty} \overline{X^r} = \sum_{r=1}^{\infty} k(P_{ns})^{r-1} = k \sum_{r=0}^{\infty} (P_{ns})^r = \frac{k}{1 - P_{ns}}.$$

Thus, X (the average number of recipients per transmission) is given by

$$X = \frac{k}{(1 - P_{ns})E[\text{Trans}]}. \quad (18)$$

Now, (11)–(13) provide an expression for $P_{ns} = 1 - P_s$ that is in terms of X , and (18) gives an expression for X in terms of P_{ns} . The two equations can be solved together to obtain P_{ns} . Once P_{ns} is obtained, (17) can be used to obtain the average number of times that a message must be transmitted in order to be received by all of the multicast group members.

Fig. 4 shows these values for a case with 100 nodes and five members per multicast group. Also shown in the figure is the lower bound on the number of required transmissions from (1). It is interesting to observe that, at least in the case examined here, the random backoff protocol only needs at most one more transmission (on average) than the lower bound. As the receiver load increases with increasing number of wavelengths, the efficiency of the random protocol also increases. More interestingly, Fig. 5 compares the transmission efficiency of the random backoff algorithm to that of the persistent algorithm. It is interesting to note that, while the algorithms exhibit similar behavior, the backoff algorithm always results in higher throughput.

IV. RECEIVER ALGORITHM

We have seen so far that the throughput of a random MSA can be improved by introducing a random delay between retransmissions of a message. An additional improvement in the performance can be achieved by having nodes intelligently, rather than randomly, select the message they receive in the event of multiple simultaneous transmissions to the same node (receiver conflict).

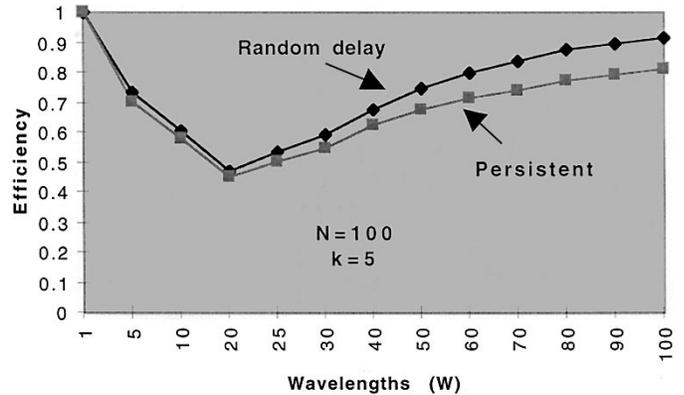


Fig. 5. Comparison of the throughput efficiency for MSA's with and without the random backoff delay.

Here we consider an algorithm where nodes select the message with the smallest number of (remaining) intended recipients. The intuition behind the algorithm is that, by selecting the message with the smallest number of intended recipients, the probability that a message will be released (having been received by all of its intended recipients) is maximized, thereby making way for the transmission of a new message. If, alternatively, a message is selected with a larger number of intended recipients, it is more likely that message will have to be retransmitted anyhow. For example, suppose a node has two messages to choose from, one which is intended for ten nodes and a second which is only intended for one node. By selecting the message intended for one node, we are assured that its channel will be freed, making way for a new multicast transmission. On the other hand, had we selected the message with ten intended recipients, it is highly likely that both messages will require retransmission.

The performance of this simple algorithm, obtained via simulation, is plotted in Fig. 6. Also plotted in Fig. 6 are the performance of random selection for the backoff algorithm (obtained via the analysis of Section III), the performance of FCFS selection for the persistent algorithm (obtained via the analysis of Section II) and the performance of random selection for the persistent algorithm (obtained via simulation). As is shown in Fig. 6, the worst performance is that of the random-selection persistent algorithm. The FCFS selection, which was analyzed in Section II, offers a small improvement. An additional improvement is obtained when we introduce the random backoff protocol. Finally, the best throughput efficiency is obtained by the random backoff protocol with the priority selection algorithm described above. In all, for the case examined here (100 nodes and $k = 5$), we see an improvement of as much as 40% in a heavily loaded system when going from the random selection persistent protocol to the priority selection random backoff protocol. However, despite its apparent advantages, the priority selection policy is inherently unfair to messages with a large multicast group size. So, while some gains may be realized in terms of overall throughput, those gains may be outweighed by the unfairness of the algorithm if this form of fairness is a concern.

In Fig. 7, we plot the multicast efficiency for the special case of $N = W$ (as in an $N \times N$ switch) with an infinite

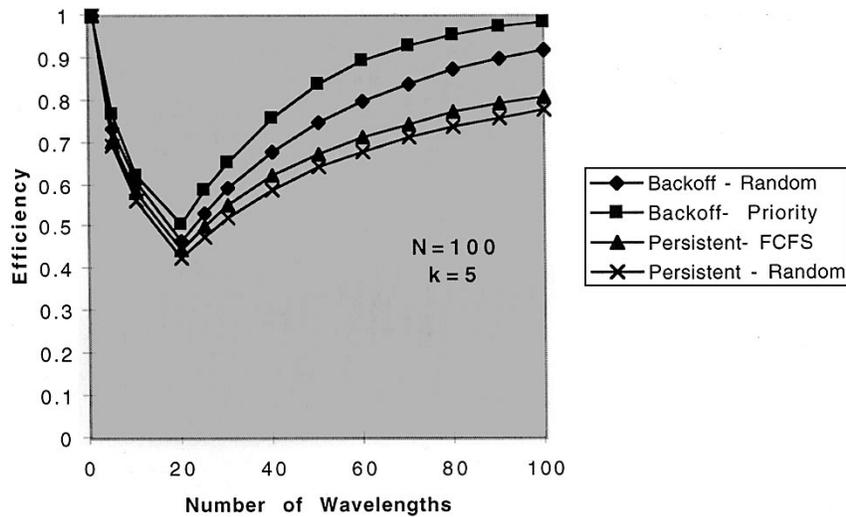


Fig. 6. Efficiency for different receiver selection algorithms.

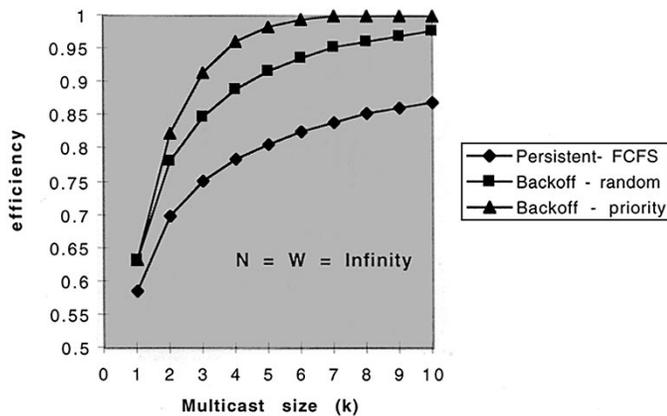


Fig. 7. Efficiency for different algorithms with infinite number of nodes and channels.

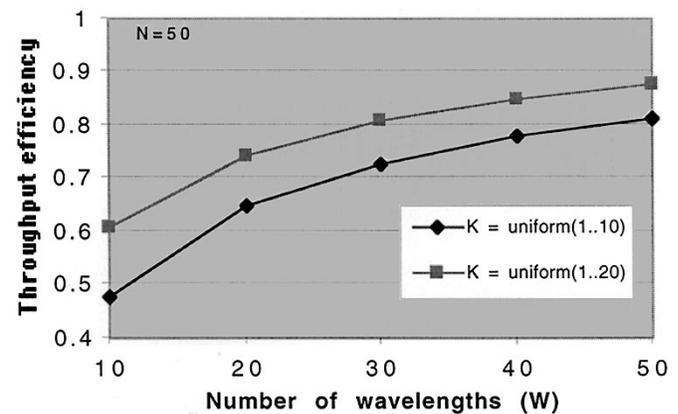


Fig. 8. Efficiency of persistent algorithm with uniformly distributed destination set size and 50 nodes.

number of nodes and wavelengths. As can be seen, the backoff algorithm reduces the effect of HOL blocking. This is a well-known result for the unicast case [13]. Here we extend this result to show that for multicast transmissions over an $N \times N$ switch utilization can be improved by introducing a form of randomization to the inputs. We also show a further improvement in utilization when receivers select the transmission to which they listen by giving priority to the message with the least number of remaining intended recipients. It is interesting to note that, for a system employing this strategy, nearly 100% utilization can be achieved when k is greater than 6.

V. GENERALIZED TRAFFIC MODELS

The analysis done so far assumed a fixed destination set size of k . Here we consider some more general distributions for k . First, we study the impact of having a uniform distribution for k . The use of a uniform distribution is consistent with the distribution used in [9] and will allow comparison to the results from [9]. To analyze the performance of our protocol under uniform distribution, we resort to simulation. Fig. 8, shows the throughput for the persistent algorithm when k

is uniformly distributed between 1 and 10 and when k is uniformly distributed between 1 and 20.

We are now able to provide a simple comparison of the algorithms presented here to the MSA from [9]. Simulation results given in [9] for a system with 50 nodes and 10 wavelengths indicate a throughput efficiency of 20% when k is uniformly distributed between 1 and 10. In contrast, Fig. 8 shows throughput efficiency greater than 40%. Similarly, when k was distributed between 1 and 20, the MSA algorithm from [9] results in throughput efficiency of 24% while Fig. 8 shows throughput higher than 60%. While one may expect that a scheduling algorithm would outperform the simple random algorithms presented in this paper, the MSA algorithm from [9] fails to do so because it requires all receivers to be available before a transmission takes place. When there are many receivers per message, this requirement significantly degrades efficiency. While the algorithms presented in this paper may not be optimal, they can be used as a benchmark for comparing the performance of other, more sophisticated, MSA's. Clearly, using a complex scheduling algorithm only makes sense if the performance of that algorithm is substantially better than of a random scheduling algorithm.

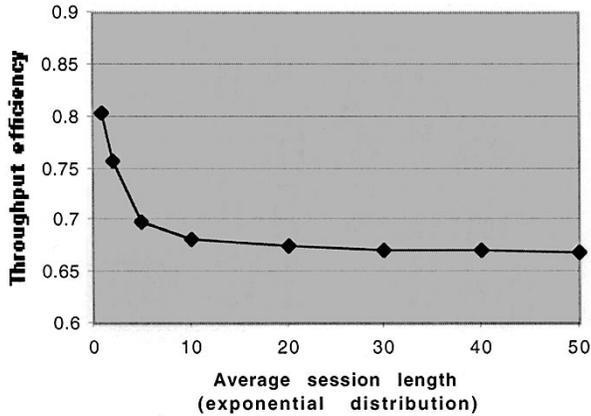


Fig. 9. Multicast efficiency for session traffic with exponentially distributed session lengths.

Another interesting generalization of the traffic model is to allow for long multicast sessions in which a node sends multiple back-to-back messages to the same multicast group. To capture this effect, we allow multicast sessions where every packet from a given session is addressed to the same multicast group. We assume that the duration (in messages) of these sessions is exponentially distributed. In this way, we allow for a mix of longer and shorter sessions. Again, we use simulation to analyze the throughput efficiency of the persistent algorithm in the presence of such multicast sessions. Fig. 9, shows the throughput efficiency for various average session lengths for a system with 100 nodes and wavelengths and destination set size uniformly distributed between 1 and 10. As can be seen, throughput efficiency degrades for session traffic. However, it appears that the drop in efficiency slows down as session length increases. Nonetheless, this does provide evidence that for session traffic a well-designed scheduling algorithm can lead to performance improvements.

VI. EFFECT OF MULTIPLE RECEIVERS PER NODE

Finally, we consider the impact of having multiple receivers/node. The motivation behind this is obvious. HOL blocking results when multiple transmissions are intended for the same node and hence only one of them can be received. This has the effect of reducing channel utilization because messages may have to be transmitted multiple times in order to be received by all of their intended recipients. With multiple receivers, nodes will be able to receive multiple transmissions simultaneously, thereby alleviating the effect of HOL blocking.

The analysis of the protocols of the previous section can be easily altered to account for the multiple receivers. The expanded analysis is omitted for brevity. Instead we briefly present some of the results. Figs. 10 and 11 show the performance of the persistent protocol with multiple receivers for a system with ten nodes and ten wavelengths. In Fig. 10, we plot channel efficiency. As can be seen from the figure, channel efficiency decreases with increasing k due to receiver contention, but increases with added receivers. This result is both clear and expected, because with more receivers the receiver contention problem is alleviated. It is also clear that

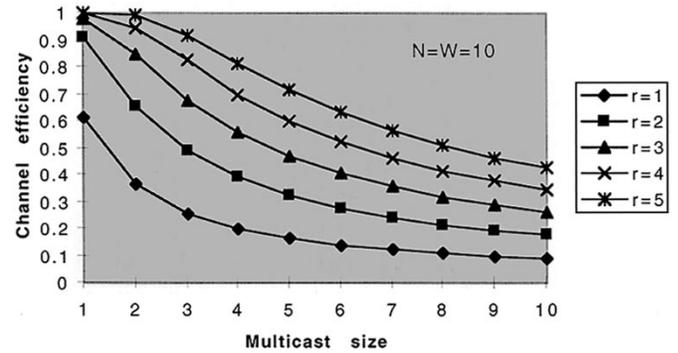


Fig. 10. Channel efficiency for persistent protocol with multiple receivers/node.

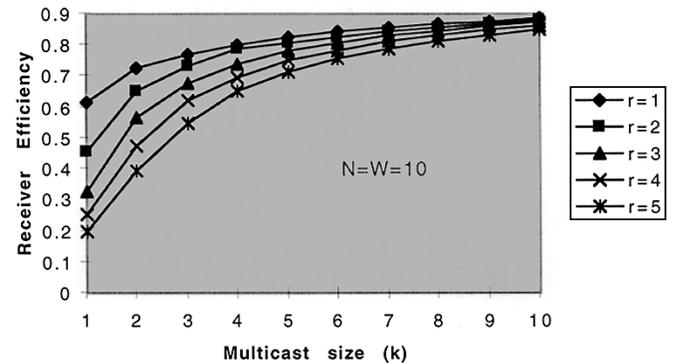


Fig. 11. Receiver efficiency for persistent protocol with multiple receivers/node.

as the number of receivers is increased the utilization of those receivers is decreased. This is shown in Fig. 11. Therefore, there appears to be a tradeoff between efficient utilization of receivers and efficient channel utilization. Increasing the number of receivers increases the channel utilization but decreases the utilization of the receivers. However, it is interesting to note that when the system is receiver limited (i.e., $kN > rW$) channel utilization can be significantly increased without a significant decrease in receiver utilization. For example, when $k = 10$, using five receivers rather than one increases channel efficiency from 10%–50%, while receiver efficiency remains very high. Similarly, the analysis for the backoff protocol yielded nearly identical results and is therefore omitted for brevity.

VII. CONCLUSION

This paper examines the performance of random multicast transmissions in WDM broadcast LAN's. This work also applies to the similar problem of multicasting in a TDM switching system. Throughput efficiency results were obtained for two random scheduling protocols, one that persistently retransmits a message until it has been received by all of the intended recipients, and another which uses a random backoff between retransmissions. Our analysis indicates that the random backoff protocol results in improved efficiency. An important observation of this paper is that the throughput degradation due to receiver conflicts is decreased as the size of the multicast group increases (as compared to the

maximum achievable throughput). Intuitively, this is because for a given number of channels, the likelihood of a receiver being idle is decreased as the number of intended recipients per transmission is increased.

We also examine the algorithms that a node uses to select which transmission to receive in the event that multiple transmissions are intended for it in the same slot. We show, through simulation, that a receiver algorithm that selects the transmission with the smallest number of intended recipients performs better over a random selection policy, albeit unfair to messages with large number of intended receivers. We also examined the benefits of having multiple receivers/node. In general, we found that increasing the number of receivers/node results in an increase in channel utilization but a decrease in receiver utilization. However, when the system is receiver limited, increasing the number of receivers can dramatically increase channel utilization while maintaining high receiver utilization.

The results of this paper are particularly applicable to systems with very fast tunable transceivers. When using slow tuning devices, the scheduling algorithms described in this paper may not perform as well since they do not attempt to take tuning times into account. Similarly, when multicast transmissions are session based, we show that throughput using the random algorithm degraded somewhat. Hence, while we conclude that simple random scheduling algorithms perform well for the system described in this paper, it is clear that different systems with different traffic types may require more sophisticated algorithms.

REFERENCES

- [1] G. J. Armitage, "IP multicasting over ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 445–457, Apr. 1994.
- [2] W. Stallings, "Ipv6: The new Internet protocol," *IEEE Commun. Mag.*, vol. 34, pp. 96–108, July 1996.
- [3] I. P. Kaminow *et al.*, "A wideband all-optical WDM network," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 780–799, June 1996.
- [4] E. Modiano, R. Barry, and E. Swanson, "Design and analysis of an asynchronous WDM local area network using a master/slave scheduler," presented at INFOCOM'99, New York, NY.
- [5] B. Mukherjee, "WDM-based local lightwave networks, Part I: Single-hop systems," *IEEE Network*, vol. 6, pp. 12–27, May 1992.
- [6] N. Mehravari, "Performance and protocol improvements for very high speed optical fiber local area networks using a passive star topology," *J. Lightwave Technol.*, vol. 8, pp. 520–530, Apr. 1990.

- [7] K. M. Sivalingam and P. W. Dowd, "A lightwave media access control protocol for WDM-based distributed shared memory system," presented at IEEE INFOCOM'96, San Francisco, CA.
- [8] G. N. Rouskas and M. H. Ammar, "Multidestination communication over tunable-receiver single-hop WDM networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 501–511, Apr. 1997.
- [9] M. S. Borella and B. Mukherjee, "A reservation-based multicasting protocol for WDM local lightwave networks," presented at the ICC'95, Seattle, WA.
- [10] J. P. Jue and B. Mukherjee, "The advantages of partitioning multicast transmissions in a single-hop optical WDM network," presented at the ICC'97, Toronto, Canada.
- [11] K. L. Yeung, K. F. Au-Yeung, and L. Ping, "Efficient time slot assignment in a TDM multicast switching system," presented at the ICC'97, Toronto, Canada.
- [12] M. K. M. Ali and S. Yang, "Performance analysis of random packet selection policy for multicast switching," *IEEE Trans. Commun.*, vol. 44, pp. 388–398, Mar. 1996.
- [13] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing in a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347–1356, Dec. 1987.
- [14] L. Kleinrock, *Queueing Systems*. New York: Wiley, vol. 1, pp. 167–226, 1976.
- [15] E. Modiano and A. Ephremides, "A method for delay analysis of interacting queues in multiple access systems," presented at the IEEE INFOCOM'93, San Francisco, CA.
- [16] G. N. Lance, *Numerical Methods for High Speed Computers*. London, U.K.: Iliffe, 1960, pp. 134–138.
- [17] J. Y. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. 5, pp. 1264–1273, Oct. 1987.



Eytan Modiano (M'90) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986, and the M.S. and Ph.D. degrees, both in electrical engineering, from the University of Maryland, College Park, in 1989 and 1992, respectively.

He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow from 1992 to 1993, while conducting research on security and performance issues in distributed network protocols. From 1993 to 1999, he was with the Communications Division of MIT Lincoln Laboratory, Lexington, MA, where he worked on communication protocols for satellite, wireless, and optical networks, and was the Project Leader for MIT Lincoln Laboratory's Next Generation Internet project. In 1999, he joined the Aeronautics and Astronautics Department at MIT, where he conducts research on communication networks and protocols, with emphasis on satellite and hybrid networks and high-speed networks. He has published over 30 papers on various aspects of data networks, including multiple access, queueing systems and distributed protocols.