Efficient Algorithms for Performing Packet Broadcasts in a Mesh Network

Eytan Modiano, *Member, IEEE*, and Anthony Ephremides, *Fellow, IEEE*

***Abstract*—We consider processors communicating over a mesh network with the objective of broadcasting information among each other. One instance of the problem involves a number of nodes all with the same message to be broadcasted. For that problem, a lower-bound on the time to complete the broadcast, and an algorithm which achieves this bound are presented. In another instance, every node in the mesh has packets to be broadcast arriving independently, according to a Poisson random process. The stability region for performing such broadcasts is characterized, and broadcast algorithms which operate efficiently within that region are presented. These algorithms involve interacting queues whose analysis is known to be very difficult. Toward that end we develop an approximation which models an n-dimensional infinite Markov chain as a single-dimensional infinite Markov chain together with an n-dimensional finite Markov chain. This approximate model can be analyzed and the results compare favorably with simulation.**

I. Introduction

A COMMON task for network protocols is the broadcasting of information from one node to the rest of the nodes in the network. This task is often required during the execution of parallel algorithms in a network of processors, or other situations where the nodes of a mesh network generate packets to be broadcast at random time instances. The latter case, which is of interest here, is motivated by the following problem: A number of satellites, laid out in space in a mesh topology, must occasionally broadcast information to the rest of the satellites. Each satellite is able to receive information from all of its neighbors simultaneously, but can only transmit in one direction at a time. This assumption follows from the use of optical beams for communication; it runs contrary to usual assumptions about wireless or cable communications.[1] However, a similar situation may also arise in wireless systems employing highly directive antennas. Furthermore, it applies to the envisioned architectures of multisatellite personal communication systems. The objective is then to develop a minimum delay algorithm for performing these message broadcasts.

Manuscript received March 1993; revised April 1995; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor N. Shacham. This work was supported by The Naval Research Laboratory and by the National Science Foundation under Grant NSF-EEC94-02384.

E. Modiano is with the Communications Division, MIT Lincoln Laboratory, Lexington, MA 02173 USA (email: modiano@ll.mit.edu).

A. Ephremides is with the Electrical Engineering Department, University of Maryland, College Park, MD 20742 USA (email: tony@eng.umd.edu).

Publisher Item Identifier S 1063-6692(96)06092-X.

[1] In the case of optical beams, each node can have individual mirrors looking at its transmitting neighbors permanently, while it must slew its transmission mirror toward each receiving neighbor separately. In this paper, for simplicity, we assume the slew delay to be negligible; however we seek algorithms which require very little slewing.

A similar problem was considered in [1], where efficient algorithms were developed for performing multiple broadcasts in a binary hypercube, rather than in a mesh. A $n \times n$ mesh is a two-dimensional (2-D) n-ary hypercube and differs from a binary hypercube in that each node has a constant number of neighbors (4), regardless of n, while in a binary hypercube the number of neighbors each node has depends on the size of the cube, a fact which alters the routing problem considerably. Therefore, while our approach to solving the mesh broadcast problem is similar to that in [1], we cannot employ algorithms similar to those proposed in [1] for the binary hypercube. At the same time, the mesh topology is essential for the multisatellite systems while the hypercube topology was mainly motivated by interconnection network applications.

To broadcast a packet, a node needs to transmit the packet along a spanning tree routed at its own location. If no interfering transmissions take place, the packet will be received by all of the nodes with a delay which depends on the selected tree and the order in which the arcs of the tree are traversed. Also, the delay encountered will be at least equal to the depth of the selected tree. This simple communication task is called a *single node broadcast*. In a $n \times n$ mesh any spanning tree has depth at least equal to $n-1$ and therefore a single node broadcast requires at least $n-1$ time units. In fact, the time to complete the broadcast depends on the tree-depth and the degree of the nodes of the tree. Thus, although there are many spanning trees of depth $n-1$ in a mesh, they are not all optimal. A simple algorithm that completes a broadcast in n time slots is as follows: We partition the mesh into vertical and horizontal rings as shown in Fig. 1.[2] The start node first transmits the packet along its vertical ring, then each node on the vertical ring transmits the packet along the horizontal ring. Since the time to cover a ring is $n/2$ slots,[3] the total broadcast time is n.

In Section II, we consider the case of *multiple "start" nodes*, that is, we assume that we have d "start" nodes all of which contain the same packet to be broadcast. Our problem here is to find a placement for the d "start" nodes, and an associated broadcast algorithm, to complete the broadcast in minimum time. We show that in that case the broadcast time is lower-bounded by $n/\sqrt{2d}$ and we provide an algorithm which meets this bound.

[2] Notice from Fig. 1 that any two nodes on the ring are at most $n/2$ hops apart. This is due to the wraparound property of the mesh where the two nodes at opposite ends of each ring are connected to each other.

[3] For simplicity of the presentation, throughout this paper we assume that n takes on even values only.

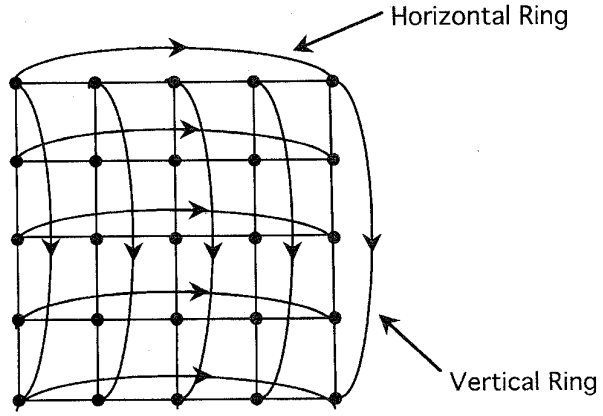


Fig. 1. A mesh with its vertical and horizontal rings.

Finally, in Section III, we consider what is the main focus of this paper, namely the case of, so called, *multiple-node broadcasts* in the mesh network. In this case, every node has its own packets to broadcast across the mesh. We assume that each node generates packets independently according to a Poisson random process of rate λ. We begin by showing that a necessary and sufficient condition for stability for multiple-node broadcasts in a mesh is given by, $\lambda \leq 1/(n^2-1)$. We then discuss broadcast algorithms that will operate within this stability region. Again, although these assumptions were motivated by multisatellite applications, they are also applicable to wireless networks with highly directive antennas which are nowadays becoming realistic alternatives for spectrally efficient indoor or cellular-like wireless systems.

A simple approach to the realization of such multiple-node broadcasts would be to require each node, when it has a packet to broadcast, to initiate a broadcast along a spanning tree routed at itself. This kind of algorithm we call an "unsynchronized algorithm." Such algorithms, despite their simplicity, do not tend to perform well in heavy traffic and are very difficult to analyze. Also, the lack of coordination between the nodes can result in an excessive need for rotating the transmitter's mirror which may result in large slewing delays. We therefore focus our attention on synchronized algorithms (algorithms where the nodes attempt to coordinate their broadcasts).

We develop two basic synchronized broadcast algorithms which attempt to minimize average delay while at the same time having a stability region equal to the one of the multiple-node broadcast as described above. Unfortunately, both algorithms involve n interacting queues which give rise to a n-dimensional Markov chain. Obtaining analytical expressions for the steady state behavior of such a system is known to be very difficult, if not impossible. Even a numerical evaluation of an n-dimensional Markov chain is computationly prohibitive [7]. This leads to the development of an approximate model for the analysis of interacting queues. The results from the approximate model compare satisfactorily with simulation, particularly when arrival rates are low.

In this paper, we do not focus on the exact sequencing of the transmissions among neighboring nodes. Rather, we try to obtain good bounds on delay performance for any such broadcast protocol. It is well known [2], [7], [8] that computing or even bounding the delay performance in interacting queue systems is a problem of formidable complexity. Thus, the value of tight bounds for design purpose is significant.

II. Multiple Start Node Broadcasts

We begin by considering the problem of performing message broadcasts in a multiple start node environment. Often it is possible to equip more than one single node with the same message in the expectation that subsequently the broadcast to the remainder of the mesh nodes will require less time to be completed. In the context of the satellite-mesh application, this situation can be achieved by a single broadcast from one or more terrestrial nodes that may reach a small subset of the mesh consisting, say, of d out of the n^2 total number of satellites. Our objective is to find a placement for the d start nodes, and an associated broadcast algorithm, so that the time to complete the broadcast is minimized. We begin by providing a lower bound on the broadcast time for a mesh with d start nodes and we go on to show an algorithm which meets this bound.

Let $\mathcal{N}$ be the set of all $n \times n$ nodes and $\mathcal{S}$ be the set of all d start nodes. Then, let $R(d)$ be the "radius" of a mesh with d start-nodes defined as follows:

$$R(d) \triangleq \max_{x \in \mathcal{N}} \left\{ \min_{s \in \mathcal{S}} \text{distance}\,(x,s) \right\}$$

where $\text{distance}(x,s)$ is the minimum number of hops needed to reach x from s or vice-versa.

So $R(d)$ is the greatest distance that any node must traverse in order to reach a start node. Clearly, it will take at least $R(d)$ time slots in order to broadcast a message throughout the entire mesh. Also for $d>1, R(d)$ will depend on where the start nodes are placed. Next we provide a lower bound on $R(d)$ which is independent of where the start nodes are located.

Proposition 1: Consider any start node x and let $n_x(i)$ be the number of nodes exactly i hops away from node x. Then

$$n_x(i) = \begin{cases} 1 & i=0 \\ 4i & 0<i<\dfrac{n}{2} \\ 4i-2 & i=\dfrac{n}{2} \\ 4(n-i) & \dfrac{n}{2}<i<n \\ 1 & i=n \end{cases}$$

Proof: For simplicity we consider only even values of n. This clearly holds for $i=0$. We next number the columns of the mesh based on their distance from the column containing node x.[4] Clearly, there is only one column of distance 0 (the one containing x); for $0<j<n/2$ there are two columns of distance j; for $j=n/2$ there is one column; and for $j>n/2$ there are no columns. Now we consider the case of $i<n/2$. When $i<n/2$ a column of distance j has exactly two nodes

[4]The distance between two columns of a mesh is the obvious distance between any node of one column and the node on the other column that lies on the same row with the first node. The distance between any two rows is defined similarly.

of distance i from x if $j<i$, and only one such node if $j=i$. It has no nodes of distance i from x if $i<j$. So the number of nodes distance i from x is equal to $\Sigma_{j=0}^{i-1}$ (number of columns of distance j) (number of nodes of distance i in column of distance j) $= 1\times 2+\Sigma_{j=1}^{i-1}(2\times 2)+2\times 1=4i$. For $i=n/2$, a column of distance 0 has one node of distance i from x. When $0<j<i$, any column of distance j has two nodes of distance i from x and when $j=i$, any column of distance j has exactly one node of distance i. So, for $i=n/2$, the number of nodes of distance i from x is equal to $1\times 1+\Sigma_{j=1}^{j=i-1}(2\times 2)+1\times 1=4i-2$. Finally, we consider the case of $n/2<i<n$ and, again, when $j<i-n/2$, column j contains zero nodes of distance i from x. When $j=i-n/2$, column j contains one node of distance i from x and when $j>i-n/2$, column j contains two nodes of distance i from x. Therefore, the number of nodes with distance i from x is equal to $2+\Sigma_{j=i-n/2+1}^{n/2-1}(2\times 2)+2=4(n-i)$. Q.E.D.

Proposition 1 tells us how many nodes are exactly i hops away from node x. We can now use Proposition 1 to determine how many nodes are within i hops of x. Let $N_x(r)$ be the number of nodes within r hops of x. Then, using Proposition 1, when $r<n/2$ we have

$$N_x(r)=1+\sum_{i=1}^{i=r}4i=2r^2+2r+1.$$

Suppose we have d start nodes. Then let $N(d,r)$ be the number of nodes within a distance r of any start node. Then clearly

$$N(d,r)\le dN_x(r)=d(2r^2+2r+1).$$

In order for every node to be within a distance r of a start node we must have

$$n^2\le N(d,r)\le d(2r^2+2r+1).$$

For this inequality to be satisfied we must have

$$0\le 2r^2+2r+\left(1-\frac{n^2}{d}\right)$$

which leads to

$$r=\frac{n}{\sqrt{2}\sqrt{d}}-1.$$

Hence, when we have d start nodes the radius must be at least $n/(\sqrt{2}\sqrt{d})$.

Theorem 1: Given a mesh with d start nodes, any broadcast algorithm must take at least

$$\frac{n}{\sqrt{2}\sqrt{d}}-1$$

time slots to complete the broadcast.

Proof: Since the radius of such a mesh must be at least $n/(\sqrt{2}\sqrt{d})-1$ links, and only one link can be traversed during one time slot, the theorem immediately follows.

Next we provide a placement for the d start nodes and an associated broadcast algorithm which meets the above bound.

When d is equal to one we have already obtained an optimal algorithm earlier. Although optimal the algorithm does not meet the above bound, therefore, the bound is not tight for all values of d; we show next that for some values of n and d an algorithm which meets the bound can be found. When d is equal to two we can place the two start nodes as far apart as possible (place one node in the center of the mesh and the other at the furthest corner) and it is easily verifiable that the radius of this topology is $n/2$. A broadcast algorithm for this topology with delay $n/2$ can now be easily realized by applying the single start node algorithm at each of the two start nodes. For d greater than two it is not as simple to find an optimal broadcast algorithm. In fact, as in the case of $d=1$, the optimal algorithm does not always meet the above bound. In [4], an algorithm was developed which came within a factor of $\sqrt{2}$ of the bound by partitioning the mesh into d square segments. Here we present an improvement to that algorithm by partitioning the mesh into diamond shaped segments.[5] The following algorithm meets the above bound for certain values of d and comes within a small factor for all other values of d. Consider values of d for which $d/2$ is a perfect square and its square root divides n. We can now partition the mesh into d diamond shaped segments as shown in Fig. 2. A start node can be placed in the center of each of these segments so that the farthest node from the start node is $n/\sqrt{2d}$ hops away. Now, each of these diamond segments can be spanned, in parallel, in $n/\sqrt{2d}$ slots and the total broadcast time for the mesh becomes $n/\sqrt{2d}$, which meets our lower bound.

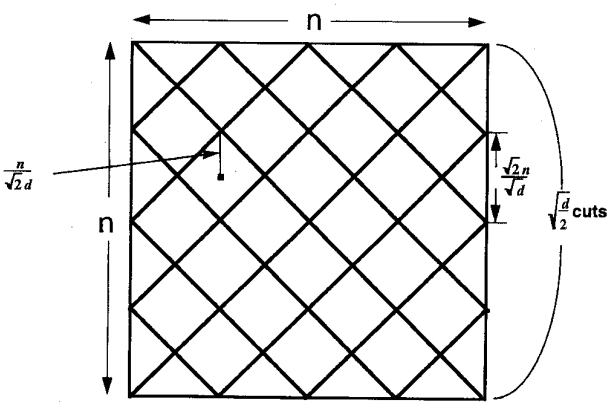


Fig. 2. A mesh partitioned into d diamond-shaped segments.

For values of $d/2$ which are not a perfect square we can always use a smaller number of start nodes, say d', for which $d'/2$ is a perfect square and $d'>d-\sqrt{2d}$. Similarly, if the $\sqrt{d/2}$ does not divide n the diamond segments will not all be of the same size, but will all have a radius $<n/\sqrt{2d}+1$.

III. Multiple-Node Broadcast Algorithms

We consider now an $n\times n$ mesh with n^2 nodes, each of which generates packets independently according to a Poisson random process of rate λ to be broadcast to the rest of the nodes in the mesh. The packets take exactly one time slot to be transmitted, each node can only transmit to one of his neighbors during a given slot, but can receive

[5] This idea is due to C. Kruskal of the Computer Science Department at the University of Maryland, College Park.

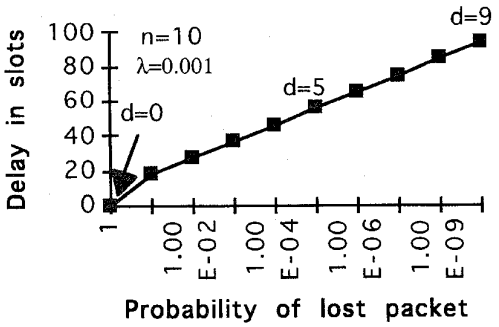


Fig. 3. Delay versus probability of lost packet (Q) with $n = 10$ and $\lambda = 0.00$.

packets from all of his neighbors simultaneously.[6] We begin by providing a necessary condition for stability for any multiple-node broadcast algorithm.

Proposition 2: In order for a multiple-node broadcast algorithm to be stable the following must hold:

$$\lambda \leq \frac{1}{n^2 - 1}.$$

Proof: The average number of packets generated in the mesh during a single time slot is λn^2. Broadcasting of any packet requires at least $n^2 - 1$ transmissions because $n^2 - 1$ nodes must receive the packet, and no transmission can be heard by more than one packet at a time. Therefore, during each slot an average demand for at least $\lambda n^2(n^2 - 1)$ packet transmissions is generated in the system. Now, since each node can only transmit in one direction at a time, at most n^2 transmissions can take place during one time slot. Therefore, for the system to be stable we must have $\lambda n^2(n^2 - 1) \leq n^2$, which proves our claim. It can be similarly shown that if nodes were allowed to broadcast along all four outgoing arcs simultaneously the requirement for stability would be $\lambda \leq 4/n^2 - 1$. Next, we will show that the condition of Proposition 2 is sufficient by providing an algorithm which is stable for all arrival rates satisfying this bound.

A. A Simple Synchronized Multiple-Node Broadcast Algorithm

In this section, we describe a method for performing multiple node broadcasts that simply serves as a means for calculating delay bounds and is not intended as a specific actual protocol for performing these broadcasts. Thus, the algorithm to be described is based on performing periodic, complete, multinode broadcasts, throughout the entire system. A multinode broadcast is the task where each node in the system broadcasts one packet (the same one) to every other node in the system. By performing periodic multinode broadcast cycles we allow each node to broadcast exactly one packet per cycle.

Proposition 3: A multinode broadcast takes $O(n^2 - 1)$ time slots to be performed.

Proof: Each node must receive $n^2 - 1$ messages, and since it has only four links on which a message can be received, our claim holds.

[6] In accordance with the nature of the satellite mesh example that requires optical links amongst the nodes.

Consider the following multinode broadcast algorithm.

Step 1) Every node broadcasts one message along its vertical ring, so that every node on the ring ends up containing all of the messages from the ring. If a node has no packets to send it sends a null packet. Similarly, if a node has more than one packet, it must wait for the next cycle before broadcasting its second packet.

Step 2) All nodes broadcast all of the messages from their vertical ring along their horizontal ring, so that all nodes in the mesh end up acquiring all of the messages transmitted.

During each slot every node across a vertical ring performs an one-packet transmission to its neighbor in the same direction, for the duration of the first step. During the second step, a variable number of slots is needed to perform similar unidirectional transfers along the horizontal rings, depending on the number of packets each vertical ring accumulates during the first step. If null packets are counted, then $n(n - 1)$ slots are needed. The incorporation of "null" or "dummy" packets in the transmission process simplifies the calculations by maintaining a constant cycle duration, but foregoes potential overall delay savings that might be realized if such dummy packet transmissions were not permitted. Thus the resulting performance would tend to be pessimistic.

1) Delay Analysis: Step 1) takes a total of $n - 1$ time slots, because all nodes can broadcast their messages in parallel along the ring. By the same reasoning, Step 2) takes $n(n - 1)$ slots. Therefore, the algorithm takes a total of $n^2 - 1$ time slots to be performed.

Our simple synchronized algorithm uses the above multinode broadcast algorithm periodically as follows. We perform a complete multinode broadcast according to the above algorithm every $n^2 - 1$ time units. If a node has no packet to send, it simply sends a null packet. This algorithm serves each node every $n^2 - 1$ time units. Therefore, the queue at each node behaves as an M/D/1 queues with synchronization. That is, we have an M/D/1 queue, where service is offered at prespecified instants of time which are $n^2 - 1$ slots apart. The delay associated with such a system is the same as the delay for an ordinary M/D/1 system with service duration $n^2 - 1$ plus the expected duration of the time elapsing between the arrival instant of some customer and the beginning of the next slot. This quantity is called the average *synchronization time* and is equal to half the service time. Therefore, the delay for this algorithm, D, is

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

Also, an M/D/1 queue with arrival rate λ and service time $n^2 - 1$ is stable if and only if

$$\lambda \leq \frac{1}{n^2 - 1}.$$

This algorithm has the same stability region as that described in Proposition 2; however, the delay associated with this algorithm is very high even for very low arrival rates. For very high arrival rates every node has a packet to send,

therefore no slots go unutilized and the algorithm is optimal. However, for low arrival rates, the algorithm yields many unused slots and results in delay of $O(n^2)$. One would expect that a good algorithm would result in $O(n)$ delay for low arrival rates, because a single node broadcast can be performed in n slots. In the next sections we consider alternative broadcast algorithms which perform as well as this algorithm for high arrival rates and result in much less delay for low arrival rates.

B. A Synchronized Multiplenode Broadcast Algorithm with Lost Packets

The previous algorithm was based on performing complete multinode broadcasts in a synchronized fashion. The problem with such algorithms is that at low traffic rates very few of the nodes have a packet to be broadcast, and therefore in performing a complete multinode broadcast many time slots are wasted. In this algorithm, we attempt to take into account the fact that at low arrival rates complete multinode broadcasts are wasteful. This algorithm performs synchronized "partial multinode broadcasts." As before, this algorithm is based on dividing the mesh into vertical and horizontal rings so that every node is associated with exactly one vertical and one horizontal ring. The algorithm is then as follows.

Step 1) As before, every node broadcasts one message along its vertical ring, so that every node on the ring ends up acquiring all of the messages from the ring.

Step 2) Every ring selects, according to some priority scheme, up to d packets to be further broadcast through the mesh. The packets not selected on that ring are discarded (clearly, actual rather than null packets are selected first; if a ring does not have enough actual packets then the remaining slots are filled with null packets). All nodes on a given vertical ring have the same d packets, to be broadcast through the mesh.

Step 3) All nodes broadcast the d "select" packets through their horizontal rings.

1) Delay Analysis: The duration of each cycle is easily computed as before. Spanning the vertical rings again takes $n-1$ slots. Spanning the horizontal rings takes an additional $d(n-1)$ slots. Therefore, the cycle length, S, is equal to $(d+1)(n-1)$.

Assuming that each node has packets arriving according to a Poisson random process of rate λ, the queue at each node behaves as a synchronized M/D/1 queue with arrival rate λ and service time $(d+1)(n-1)$. Notice that this is so regardless of the scheme used to select the d packets. This is so because at every cycle, the queue at each nonempty node is reduced by one. The only effect that the priority scheme has is to determine which of the messages actually reaches the rest of the nodes and which are permanently lost. The delay for this queue is simply

$$\text{Delay} = \left(\frac{3}{2} + \frac{\lambda S}{2(1-\lambda S)}\right) S$$

where S is the cycle length and is equal to $(d+1)(n-1)$. Of course such a queue is stable as long as $\lambda \leq 1/[(d+1)(n-1)]$. Notice, that if we select d equal to n what we have is the complete multinode broadcast algorithm of the previous section. Also, notice that delay is minimized when d is equal to one. However, the drawback of this algorithm is that excess packets are lost. That is, if a ring has more than d nonempty nodes, the packets which are not selected are forever lost.[7] Clearly, if d is taken equal to n, no packets will be lost, but delay could be intolerable for low arrival rates. If d is taken to be one, delay is minimized, but the probability of a lost packet could be high. Next we evaluate the probability of a lost packet.

2) Probability of Lost Packets Analysis: In order to compute the probability of a packet being lost [i.e., a packet is lost if it is at the top of its node's queue at Step 1), but does not get selected at Step 2] we must decide on a priority scheme for selecting the d packets to be served. For computational simplicity we assume that the d packets are chosen at random. Now the probability of a lost packet can be expressed as the ratio of the average number of lost packets on the ring to the average number of nonempty nodes on the ring during a single cycle (in the steady-state).

Consider an arbitrary (vertical) ring with n nodes.

Let N be the total number of packets on the ring at the beginning of a cycle in the steady state, that is the number of nonempty nodes on the ring and L be the number of lost packets during the sycle. Clearly L is equal to the maximum of $N-d$ and zero.

Then

$$E[N] = \sum_{i=1}^{n} iP(N=i)$$

and

$$E[L] = \sum_{i=d+1}^{n} (i-d)P(N=i).$$

It is easy to show that in the steady state the distribution of N is given by [3]

$$P(N=i) = \binom{n}{i}(\lambda S)^i(1-(\lambda S))^{n-i}. \tag{1}$$

Finally, Q, the steady-state probability that a packet is lost can be expressed as

$$Q = \frac{E[L]}{E[N]}. \tag{2}$$

In Fig. 3 we plot delay versus Q for $n=10$ and $\lambda = 0.001$. Each of the delay values on the horizontal axis corresponds to a different value of d. These d values are marked on the curve. When $d=n$ the algorithm performs "complete" broadcasts and no packets are lost. Similarly, when $d=0$, all of the packets are lost. Of course, the delay values are very high when $d=n$. For smaller delays we must use smaller values for d and sustain a larger probability of losing a packet.

[7] Or could be rescheduled for retransmission at a later time; however, we do not address the question of retransmissions in this paper.

C. A Synchronized Multiple Node Broadcast Algorithm with No Lost Packets

The algorithm described in the previous section is encouraging in that it shows that multiple-node broadcasts can be performed without having to sustain the unacceptable delay of a complete multinode broadcast. The major drawback of the algorithm is that it allows the loss of packets. However, it is evident from the results of the previous section that for low arrival rates the probability of a lost packet is very low. This suggests that if, instead of allowing packets to be lost, we let packets which were not selected for transmission rejoin their queues, the overall affect on delay may be minimal. We are therefore led to considering the following protocol:

Step 1) As before, every node broadcasts one message along its vertical ring, so that every node on the ring acquires all of the messages generated from that ring.

Step 2) Every ring selects, according to some priority scheme, up to d packets to be further broadcast through the mesh. The un-selected packets rejoin their nodes queues and re-attempt transmission during the next cycle. (If a ring has fewer than d packets then the remaining slots are filled with null packets.) Clearly, all nodes on a given vertical ring have the same d packets to be broadcast through the mesh.

Step 3) All nodes broadcast the d "select" packets through their horizontal rings.

Since there are n nodes on the ring and up to d of them can receive service during a cycle of duration $(d+1)(n-1)$, in order for the algorithm to be stable we must have $\lambda \leq d/[n(d+1)(n-1)]$. Clearly, for $d<n$ the stability region of this algorithm is smaller than the stability region of the mesh described in Proposition 2. However, this algorithm can accommodate all admissible arrival rates by increasing the value of d, since with $d = n$ the algorithm has the same stability region as that of the mesh.

Although this algorithm presents only a slight modification to that of the previous section, that modification makes the algorithm very difficult to analyze. In the previous algorithm analysis was simple because whether or not a node's packet was served, the node's queue was reduced by one (at the possible cost of a lost packet). This resulted in essentially n independent queues each operating as an M/D/1 system whose analysis is well known. By allowing the unserved packets to rejoin the queues, the n queues become highly dependent on one another. This is because the event of one queue getting service is dependent on whether or not the other queues in the system were served. It is also obvious that the performance of this algorithm depends on how the queues to be served are selected.[8] For the purpose of simplifying the analysis of this scheme we assume, as we did in the previous section, that the d packets are chosen at random.

[8] A desirable service policy is one that would minimize delay. We believe that that policy is the one which at each cycle serves the d largest queues. However, proof of that conjecture is by no means obvious.

We would like to compute the average delay in this system. Because of the dependence between the n queues the queue sizes in a ring of n nodes form an n-dimensional infinite Markov chain. Obtaining closed form expressions for the steady-state behavior of such a system is generally very difficult. Even a numerical evaluation of such systems can be very computationally complex [7].

A similar situation arises in the analysis of a buffered ALOHA system. In [2] and [8], approximate models for the analysis of ALOHA were developed. The basic idea behind these approximations was to split the system into two Markov chains. One single-dimensional, infinite, chain tracking the state of a single user and the other a n-dimensional, finite, Markov chain tracking the state of the rest of the system.

Before we begin with the approximation, we consider a special case of the problem where an exact solution is attainable. This is the case when d is equal to one. We can obtain the solution for the case of $d = 1$ by considering the state of the entire ring. Clearly, for the entire ring with $d = 1$, at each cycle the number of packets on the ring is reduced by one as long as the ring is not empty. Also since the ring contains n nodes each with independent Poisson arrivals of rate $\lambda(d+1)(n-1) = \lambda 2(n-1)$ packets per cycle, packets arrive at the ring according to a Poisson process of rate $2\lambda n(n-1)$. Therefore the entire ring behaves as an M/D/1 queue with synchronization and with service time $2(n-1)$ and arrival rate $2\lambda n(n-1)$. Of course, the average delay for this queue is well known and given by

$$\text{Delay}(d=1) = 2(n-1)\left(\frac{3}{2} + \frac{2\lambda n(n-1)}{2(1-2\lambda n(n-1))}\right).$$

Unfortunately, a similar approach can not be taken for values of d which are greater than one (except $d = n$ which was solved in the previous section). This is because for other values of d the entire ring is no longer a simple M/D/1. In fact it is no longer M/D because, since only one packet is allowed per node during a cycle, service will depend on the way packets are distributed among the nodes and is not deterministic. In what follows we consider an approximation for the n-dimensional infinite Markov chain which is based on the two-chain model described earlier.

1) The Two-Chain Model: The idea behind having two Markov chains, one for a single user and the other for the rest of the nodes, is that while a single infinite chain with n dimensions is very difficult to analyze, each of the two smaller chains are analyzable and when solved together provide an approximation to the original chain. The one-dimensional (1-D) infinite user-chain represents the buffer size and state for a single user and can be solved by conditioning on the state of the system-chain. In turn, the system-chain tracks the interaction between the different users.

2) The User's Markov Chain: The user's Markov chain represents the queue size for a single user. It is therefore an infinite chain. Fig. 4 shows the user-chain. Arrivals are Poisson and departures are geometrically distributed with parameter P_s representing the probability that a node is "selected" during a given cycle. Thus, the user-chain is an M/G/1 with geometrically distributed service time.

Fig. 4. The user's chain.

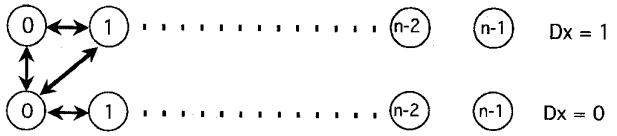


Fig. 5. The system-chain.

This is an approximation to the real system because we assume that the probability of a successful transmission is independent of the number of packets in the queue. The average queue size, $\overline{n}_x$, for this system can be easily computed using the well known formulas for an M/G/1 and the first and second moments of the geometric distribution, namely

$$\overline{n}_x = \frac{\lambda S(2-\lambda S)}{2(P_s-\lambda S)} \tag{3}$$

and the average delay is obtained using Little's formula

$$\text{delay} = \frac{S}{2} + \frac{\lambda S(2-\lambda S)}{2\lambda(P_s-\lambda S)} \tag{4}$$

where S is the cycle duration and is equal to $(d+1)(n-1)$ and the term $S/2$ accounts for the synchronization delay of one-half cycle.

This completes the analysis of the user's chain. The only missing ingredient, in order to compute the delay, is P_s. This is the one term that we will obtain from the system-chain. Before we turn our attention to the system-chain, there is one parameter from the user-chain which will be needed later for the analysis of the system-chain. We will need to know the probability that there is exactly one packet in the queue. This is easily derived as follows, let $\Pi(i)$ be the probability that the user's system is in state i. Then,

$$\Pi(0) = 1 - \frac{\overline{A}_x}{P_s} = 1 - \frac{\lambda S}{P_s}$$

We can next express $\Pi(1)$ in terms of $\Pi(0)$ by writing the steady-state flow equation out of state zero. So

$$\Pi(0) = A(0)\Pi(0) + A(0)P_s\Pi(1)$$

where $A(0)$ is the probability of having no arrivals during a slot, and is equal to $\exp(-\lambda S)$. So,

$$\Pi(1) = \frac{\Pi(0)(1-A(0))}{A(0)P_s} = \frac{P_s - \lambda S(1-A(0))}{A(0)P_s^2}.$$

2) The System Markov Chain: In [3], a number of approaches were considered for the system chain. The most promising of which was a system chain model that merely represents the identity and state of one user along with the number of nonempty nodes on the ring. Requiring the system-chain to contain the state of a single user allows us to more accurately derive the probability of success for the user-chain. This is because the probability of success is defined to be the probability that the user is chosen given that it has a packet to send. Therefore, when the system-chain contains the state of our user, we can compute the probability of success by conditioning on the user-state being nonempty. The system-chain will include information on whether or not node x is empty as well as the total number of empty nodes in the system. Let the pair (w, D_x) represent the state of the system, where w equals the number of nonempty nodes and D_x is equal to zero if node x is empty and one otherwise. Clearly, there are a total of $2n$ possible states. Fig. 5 shows the system-chain.

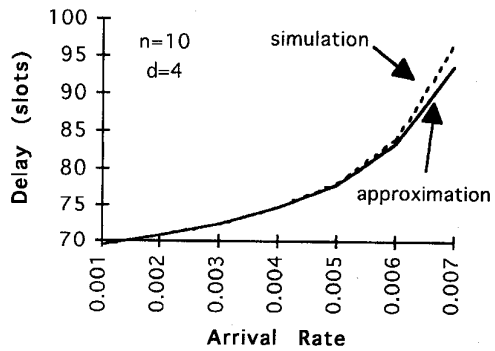


Fig. 6. Delay versus arrival rate (λ) with $n = 10$ and $d = 4$.

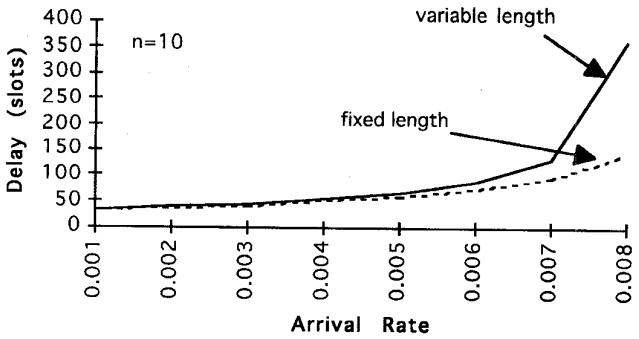


Fig. 7. Delay versus arrival rate (λ) with $n = 10$.

In order to express the transition probabilities for this chain we define the following two quantities. Let $B_{(i,j)}(k)$ be the probability that k empty nodes become busy (nonempty) given that we started in state (i,j) and let $C_{i,j}(k,l)$ be the probability that k busy nodes become empty, and that D_x goes from j to l. We can now write the probability of going from state (i,j) to (i',j') as follows:

$$P((i,j),(i',j')) = \sum_{k=0}^{k=n} B_{(i,j)}(k)C_{i,j}(k-(i'-i),j').$$

The term $B_{(i,j)}(k)$ does not depend on the state of node x and is simply the probability that k of the $n-i$ empty nodes receive packets. Thus

$$B(i,j)(k) = \begin{cases} \binom{n-i}{k}(1-A(0))^k A(0)^{n-i-k}, & \text{if } k \le n-i \\ 0, & \text{otherwise.} \end{cases}$$

The term $C_{i,j}(k,l)$ does depend on the state of node x. Also, it depends on P_e, i.e., the probability that a node

becomes empty upon receiving service. That probability can be expressed in terms of the user-chain steady-state probabilities as follows:

$$P_e = \frac{\Pi(1)A(0)}{1-\Pi(0)}$$

where the numerator is the probability that the node had one packet in the queue and received no new packets while the denominator represents the probability that the queue was not empty (a necessary condition in order for the node to receive service).

In order to compute $C_{i,j}(k,l)$, we consider four cases representing the possible combinations of values of j and l. Also, we define the following three term that will help simplify the expressions:

$$Z1 = \binom{d}{k} P_e^k (1-P_e)^{d-k}, \quad Z2 = \binom{d-1}{k} P_e^k (1-P_e)^{d-1-k}$$

and

$$Z3 = \frac{i+1-d}{i+1} Z1 + (1-P_e)\frac{d}{i+1} Z2.$$

The term $Z1$ represents the probability that when d nodes are served k of them become empty. The $Z2$ term represents the probability that when $d-1$ nodes are served k of them become empty, and $Z3$ is the probability that when d nodes are served and node x is full k of the nodes become empty and node x remains full. Of course, all of these quantities are approximate in that we assume that, P_e, the probability that a node becomes empty upon receiving service is independent of the state of the system.

Case 1: $j = 0$ and $l = 0$
$C_{i,0}(k,0)$ is equal to the probability of having no new arrivals times the probability that k nodes become empty given start state (i,j). When $k \le d \le i$ exactly d nodes are served and

$$C_{i,0}(k,0) = A(0)\binom{d}{k} P_e^k (1-P_e)^{d-k}.$$

When $k \le i \le d$ exactly i nodes are served and

$$C_{(i,0)}(k,0) = A(0)\binom{i}{k} P_e^k (1-P_e)^{i-k}.$$

Case 2: $j = 0$ and $l = 1$
$C_{i,0}(k,1)$ is equal to the probability of having at least one new arrival times the probability that k nodes become empty. When $k \le d \le i$ exactly d nodes are served and

$$C_{i,0}(k,1) = (1-A(0))\binom{d}{k} P_e^k (1-P_e)^{d-k}.$$

When $k \le i \le d$ exactly i nodes are served and

$$C_{(i,0)}(k,1) = (1-A(0))\binom{i}{k} P_e^k (1-P_e)^{i-k}.$$

When $j = 1$ and $l = 0, C_{i,j}(k,l)$ is the probability that k nodes become empty and node x becomes empty as well. Finally, when $j = l = 1, C_{i,j}(k,l)$ is the probability that k nodes become empty and node x remains full.

Case 3: $j = 1$ and $l = 0$
$C_{i,j}(k,l)$ is the probability that k nodes become empty and node x becomes empty as well.
If $k < d \le i$ then exactly d nodes are served and $C_{(i,1)}(k,0) = (d/i+1)P_e Z2$, corresponding to the probability that node x is served and becomes empty, and that k of the remaining $d-1$ nodes which are served become empty. Similarly, when $k \le i < d$, exactly i nodes are served and

$$C_{(i,1)}(k,0) = P_e\binom{i}{k} P_e^k (1-P_e)^{i-k}.$$

Case 4: $j = 1$ and $l = 1$
$C_{i,j}(k,l)$ is the probability that k nodes become empty and node x remains full.
Again have two cases, if $k \le d \le i$ then d nodes are served and $C_{(i,1)}(k,1) = Z3$ and if $K \le i < d, i$ nodes are served and node x is also served and

$$C_{(i,1)}(k,1) = (1-P_e)\binom{i}{k} P_e^k (1-P_e)^{i-k}.$$

Of course when i, k, and d lay outside of these boundaries $C_{(i,j)}(k,l) = 0$. Putting together the four cases we have

$$C_{(i,0)}(k,0) = \begin{cases} A(0)\binom{d}{k} P_e^k (1-P_e)^{d-k}, \\ \quad \text{if } k \le d \le i \\ A(0)\binom{i}{k} P_e^k (1-P_e)^{i-k}, \\ \quad \text{if } k \le i \le d \\ 0, \quad \text{if } j > d \text{ or } j > i \end{cases}$$

$$C_{(i,0)}(k,1) = \begin{cases} (1-A(0))\binom{d}{k} P_e^k (1-P_e)^{d-k}, \\ \quad \text{if } k \le d \le i \\ (1-A(0))\binom{i}{k} P_e^k (1-P_e)^{i-k}, \\ \quad \text{if } k \le i \le d \\ 0, \quad \text{if } k > d \text{ or } k > i \end{cases}$$

$$C_{(i,1)}(k,0) = \begin{cases} \frac{d}{i+1} P_e\binom{d-1}{k} P_e^k (1-P_e)^{d-1-k}, \\ \quad \text{if } k < d \le i \\ P_e\binom{i}{k} P_e^k (1-P_e)^{i-k}, \\ \quad \text{if } k \le i < d \\ 0, \quad \text{if } k \ge d \text{ or } k > i \end{cases}$$

$$C_{(i,1)}(k,1) = \begin{cases} \frac{i+1-d}{i+1} P_e\binom{d}{k} P_e^k (1-P_e)^{d-k} \\ \quad +(1-P_e)\frac{d}{i+1}\binom{d-1}{k} \\ \quad \cdot P_e^k (1-P_e)^{d-1-k}, \\ \quad \text{if } k \le d \le i \\ (1-P_e)\binom{i}{k} P_e^k (1-P_e)^{i-k}, \\ \quad \text{if } k \le i < d \\ 0, \quad \text{if } k > d \text{ or } k > i \end{cases}$$

These equations completely specify the transition probabilities for the system-chain. Next we need to compute P_s, i.e., the probability of node x having a successful transmission. We have

$$\begin{aligned} P_s &= P(\text{success}|D_x = 1) \\ &= \frac{P(\text{success}, D_x = 1)}{P(D_x = 1)} \\ &= \frac{\sum_{i=0}^{i=d-1} P_{i,1} + \sum_{i=d}^{i=n-1} \frac{d}{i+1} P_{i,1}}{\sum_{i=0}^{i=n-1} P_{i,1}} \end{aligned}$$

where $P_{i,j}$ is the probability of being in state (i, j). The numerator represents the probability of success when the system-chain is in one of the states with node x full. The denominator represents the probability of the system being in one of these states.

We have now expressed the system-chain in terms of the user-chain parameter P_e and the user-chain in terms of the system-chain parameter P_s. The two chains can now be solved together using an iterative algorithm to obtain the results for the approximation.

We compared the delay results predicted by the approximation to those obtained through the use of simulation. In general the approximation works well for low arrival rates and deteriorates as the arrival rate approaches saturation [3]. In figure [6] we plot Delay vs. λ with $n = 10$ and $d = 4$. As can be seen from these figures the approximation yields results that are very close to those obtained by simulation, especially for low arrival rates. Additional plots for different values of n and d are given in [3] and show similar results.

D. A Synchronized Multiple Node Broadcast Algorithm with Variable Cycle Length

The different broadcast algorithms presented in this paper attempt to minimize delay by limiting the number of idle slots during a cycle due to low traffic loads. In the previous section we described a broadcast algorithm which attempted to minimize the number of idle slots by limiting the number of nodes which can broadcast during a cycle, and keeping short, fixed length cycles. In this section we describe an alternative algorithm, which allows the cycle length to vary, and lets every node broadcast a packet during each cycle. The algorithm works as follows:

- Step 1) As before, every node broadcasts one message along its vertical ring, so that every node on the ring acquires all of the messages generated from that ring.
- Step 2) After completing step one, every node broadcasts all of the packets from its vertical ring along its horizontal ring, so that every node in the mesh contains every packet.
- Step 3) When a node finishes its transmission it sends one "empty" packet for synchronization.
- Step 4) A new cycle begins after all packets have been broadcast.

Clearly, the cycle length for this algorithm depends on the number of nodes which have a packet to send.

Let S be the number of slots required to complete the broadcast of all packets during a cycle in th esteady state and let N be the number of nodes which have a packet to send at the beginning of that cycle. Then

$$S \leq 2n + N.$$

This follows from the need to spend $n - 1$ slots traversing the vertical rings. Then the circling around the horizontal rings requires no more than $N + n$ slots, where the n slots are for synchronization. Therefore, we let the cycle length be $2n+N$. Every node will have received every packet within that time. All nodes will know the value of N, and synchronization will be maintained. Theoretically, we can allow the cycle lengths to be a little shorter (since the above expression is an upper bound on S); however, both for simplicity of the algorithm and its analysis we choose the cycle length to be $2n + N$.

The analysis of this algorithm again involves analyzing interacting queues since arrivals at the different queues depend on cycle length and those depend on the status of all n^2 queues. As in the previous section we resort to an approximate analysis in which we assume (incorrectly) that the duration of each cycle is independent of the exact number of packets queued up at one particular node.

The complete analysis is presented in [3]. Here we merely present the delay results predicted by the analysis. Let, n_x be the number of packets queued up at an arbitrary node node x at the beginning of a cycle in the steady state. And let, D_x be the indicator function for n_x, i.e., $D_x = 1$ if $n_x > 0$ and zero otherwise. It is clear that in the steady state

$$E[S] = 2n + n^2 \overline{D}_x$$

and by Little's law we know that $\lambda E[S] = \overline{D}_x$. Therefore

$$\overline{D}_x = \frac{2\lambda n}{1 - \lambda n^2}.$$

It was shown in [3] that

$$\overline{n}_x \approx \frac{\overline{A}_x^2 + \overline{D}_x(1 - 2\lambda[(2n+1) + (n^2-1)\overline{D}_x])}{2[1 - \lambda[(2n+1) + (n^2-1)\overline{D}_x]]}. \quad (5)$$

The average delay, D, can now be expressed using Little's formula, as

$$D = \frac{\overline{n}_x}{\lambda} + \frac{2n + n^2\overline{D}_x}{2}$$

where the second term represents the synchronization delay of half a cycle. Also, since $S = 2n + n^2\overline{D}_x$, under a heavy load the cycle length would be $2n + n^2$; in order for the algorithm to be stable we must have

$$\lambda \leq \frac{1}{2n + n^2}. \quad (6)$$

This approximation compares extremely well with simulation, even when arrival rates are very close to saturation [3]. In Fig. 7, we plot delay versus arrival rate when $n = 10$. Also in Fig. 7 we plot, for the purpose of comparison, the delay resulting from the fixed length algorithm of the previous

section. The algorithm of the previous section requires that we choose a value for d. Plotted in Fig. 7 are the delays resulting from use of the value of d that minimizes the delay of the algorithm. Hence the comparison is of the variable length algorithm to the fixed length algorithm with the optimal d. As can be seen from Fig. 7, under heavy load the fixed length algorithm results in lower delays than the variable length algorithms. This is due to the fact that the stability region for the variable length algorithm is slightly reduced due to the $2n$ slots used to maintain synchronization. This is in contrast to the fixed length algorithm that under heavy load uses a value of d equal to n and consequently obtaining a stability region that is equal to the upperbound given by Proposition 2.

IV. Conclusion

This paper presents broadcast schemes for a mesh network with somewhat unusual properties which resulted from a specific application of satellite broadcasts but with potential applicability to future LEO satellite systems as well as wireless networks with highly directive antennas. An interesting extension to this work would be to consider a more "typical" mesh topology, where all nodes can transmit to all of their neighbors simultaneously but only receive from one neighbor at a time. Work similar to that of this paper was done for a binary hypercube in [1]. Comparing the performance of these two topologies when the same number of nodes are involved would also be interesting. The algorithms described in this paper are vulnerable to node failures. Many applications require algorithms which can withstand failures. Inevitably, building security into a routing algorithm will result in additional delays due to redundancy. The development of such algorithms, and the analysis of the tradeoffs between delay and additional security, is also an interesting area for future research. Finally, the model developed here for analyzing interacting queues can be useful for the analysis of other systems of interacting queues. It was motivated by similar models used for analyzing the ALOHA multiple access protocol and offers an improvement over those models in the way the interaction between the user and system chain is tracked. In [6], it was shown how this model can be applied to the ALOHA protocol, and it was shown that this model can be used to effectively approximate the performance of the ALOHA protocol at a relatively low computational complexity. We believe that this model may prove to be useful in many other systems of interacting queues and in particular for multiple access schemes. It is of interest to find other systems with dependent queues, not necessarily involving multiple access, for whose analysis this model or a two chain model in general may prove to be useful.

References

[1] G. Stamoulis and J. Tsitsiklis, "Efficient routing schemes for multiple broadcasts in hypercubes," *Proc. 29th CDC*, Honululu, Hawaii, Dec. 1990, pp. 1349–1354.

[2] T. N. Saadawi and A. Ephremides, "Analysis, stability, and optimization of slotted ALOHA with a Finite number of buffered users," *IEEE Trans. Automat. Contr.*, June 1981.

[3] E. Modiano, "Security and performance issues in distributed protocols," Ph.D. dissertation, Department of Electrical Engineering, The University of Maryland, College Park, MD, 1992.

[4] E. Modiano and A. Ephremides, "Efficient routing schemes for multiple broadcasts in a mesh," in *26th Annu. Conf. Information Sciences and Systems,* Princeton, NJ, Mar. 1992.

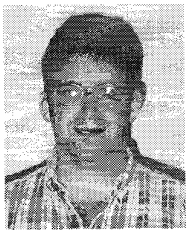
[5] E. Modiano and A. Ephremides, "The performance of a multiple node broadcast algorithm: An analysis of interacting queues," in *1993 IEEE Int. Symp. Inform. Theory*, San Antonio, TX, Jan., 1993.

[6] E. Modiano and A. Ephremides, "A method for delay analysis of interacting queues in multiple access systems," in *INFOCOM 1993*, San Francisco, CA, Apr. 1993.

[7] T. Nakasis and A. Ephremides, "Steady-state behavior of interacting queues—A numerical approach," *IEEE Trans. Inform. Theory*, Mar., 1990.

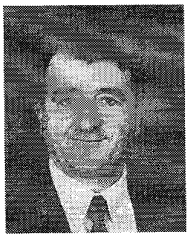
[8] A. Ephremides and R. Z. Zhu, "Delay analysis of interacting queues with an approximate model" *IEEE Trans. Commun.*, Feb., 1987.

[9] G. N. Lance, *Numerical Methods for High Speed Computers.* London, U.K.: Iliffe, 1960, pp. 134–138.



Eytan Modiano (M'90) received the B.S. degree in electrical engineering and computer science from the University of Connecticut, Storrs, in 1986, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, MD, both in electrical engineering, in 1989 and 1993, respectively.

He was a Naval Research Laboratory Fellow between 1987 and 1993 and a National Research Council Post Doctoral Fellow during 1993–1994 while he was conducting research on security and performance issues in distributed network protocols. Presently, he is with the Communications Division of MIT Lincoln Laboratory, Lexington, MA, where he is studying various aspects of data networking over satellites and internetworking of satellite and terrestrial networks. His research interests are in the areas of communication networks and protocols, multiple access protocols, queueing systems, and satellite communications.



Anthony Ephremides (F'84) received the B.S. degree from the National Technical University of Athens, in 1967, and the M.S. and Ph.D. degrees from Princeton University, Princeton, NJ, in 1969 and 1971, respectively, all in electrical engineering.

He has been at the University of Maryland since 1971, and currently holds a joint appointment as Professor in the Electrical Engineering Department and the Institute of Systems Research (ISR). He is co-founder of the NASA Center for Commercial Development of Space on Hybrid and Satellite Communications Networks established in 1991 at Maryland as an off-shoot of the ISR. He was a Visiting Professor in 1978 at the National Technical University in Athens, Greece, and in 1979 at the EECS Department of the University of California, Berkeley, and at INRIA, France. During 1985–1986, he was on leave at MIT and ETH in Zurich, Switzerland. He was the General Chairman of the 1986 IEEE Conference on Decision and Control in Athens, Greece. He has also been the Director of the Fairchild Scholars and Doctoral Fellows Program, an academic and research partnership program in Satellite Communications between Fairchild Industries and the University of Maryland. He has been the President of the Information Theory Society of the IEEE (1987), and served on the Board of the IEEE (1989 and 1990). His interests are in the areas of communication theory, communication systems and networks, queueing systems, signal processing, and satellite communications.

Dr. Ephremides received the IEEE Donald E. Fink Prize Paper Award in 1992.