Efficient Routing Algorithms for Multiple Broadcasts in a Mesh

Eytan Modiano and Anthony Ephremides
Electrical Engineering Dept.
University of Maryland
College Park, MD

Abstract

We consider processors communicating over a mesh network with the objective of broadcasting information amongst 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 very well with simulation.

1 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. In this paper we consider a situation where the nodes of a mesh network generate packets to be broadcast at random time instances. We were motivated by the following problem: A number of satellites, laid out in space in a mesh topology are required to broadcast information amongst each other. Each satellite is able to receive information from all of its neighbors simultaneously, but can only transmit in one direction at a time. [1] The objective of that problem was to develop a delay optimal algorithm for performing these message broadcasts. A similar problem was considered in [1] where efficient algorithms were developed for performing multiple broadcasts in a binary hypercube.

In order to broadcast a packet, all that a node needs to do is 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 an n × n, mesh any spanning tree has depth at least equal to n-1 and therefore a single node broadcast must take 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 Figure 1. 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, the total broadcast time is n.

In section 2 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 this case the broadcast time is lower-bounded by $\frac{n}{\sqrt{2d}}$ and we provide an algorithm which meets this bound.

Finally, in section 3, 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, $\lambda \leq \frac{1}{n^2-1}$. We then discuss broadcast algorithms that will operate within this stability region.

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. 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 computationally prohibitive [4]. This leads us to the development of an approximate model for the analysis of interacting queues. The results from our approximate model com-

[1] This assumption follows from the use of optical beams for communication; it runs contrary to usual assumptions about wireless or cable communications.

pare very well with simulation, particularly when arrival rates are low [3].

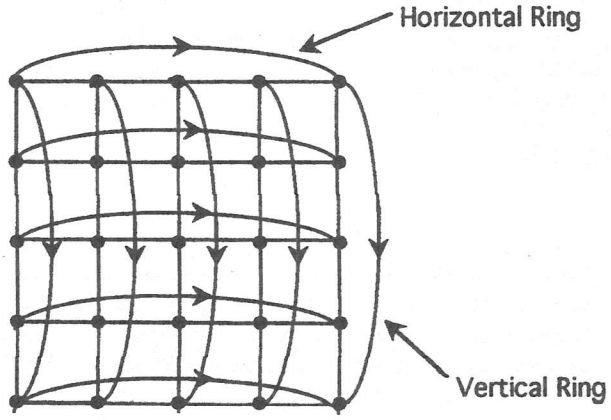

Figure 1. A mesh with its vertical and horizontal rings.

2 Multiple Start node broadcasts

We now turn our attention to the more interesting task of multiple "start" nodes. Let there be d such nodes. That is we can choose d nodes on the mesh which will all have the same message to be broadcast. 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 $R(d)$ be the radius of a mesh with d start nodes, defined as follows:

$$R(d) = \max_{\text{all nodes } x} \left\{ \min_{\text{all start nodes } s} distance(x,s) \right\}.$$

Where distance is measured in number of "hops". 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 [5] *Consider any start node x and let $n_x(i)$ be the number of nodes exactly i hops away from node x. Then,*

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

Proposition 1 tells us how many nodes are exactly i hops away from node x. It follows directly from the mesh topology. We can now use this property 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, we have

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

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

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

For this inequality to be satisfied we must have,

$$0 \leq 2r^2 + 2r + (1 - \frac{n^2}{d}).$$

Which, in turn, implies that

$$\begin{aligned} r &\geq \frac{-2 \pm \sqrt{4 - 8(1 - \frac{n^2}{d})}}{4} \\ &\geq \frac{n}{\sqrt{2}\sqrt{d}} - 1 \end{aligned}$$

So, when we have d start nodes the diameter must be at least $n/(\sqrt{2}\sqrt{d}) - 1$.

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 diameter 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 which requires n slots. 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 diameter of this topology is $n/2$. For d greater than 2 it is not as simple to find an optimal broadcast algorithm. The following algorithm approaches our bound within a constant multiplier. For simplicity, we assume that d is a perfect square and that n divides the square root of d. We can now partition an $n \times n$ mesh into d square segments each with diameter $n/\sqrt{d}$. This is done simply by dividing the mesh along its vertical and horizontal dimensions into $\sqrt{d}$ parts each of size $n/\sqrt{d}$. The result is d square segments of size $n/\sqrt{d} \times n/\sqrt{d}$. We can now place the d start node in the center of each of these squares (as was done in the single start node example). The message can now be broadcast in each of these smaller mesh segments using the optimal algorithm for the single start node case. Since the d smaller meshes can be handled in parallel, each requiring $n/\sqrt{d}$ slots, the total broadcast time for the mesh with d start nodes will be $n/\sqrt{d}$ slots. Figure 2 describes the partitioning of the mesh into d meshes.

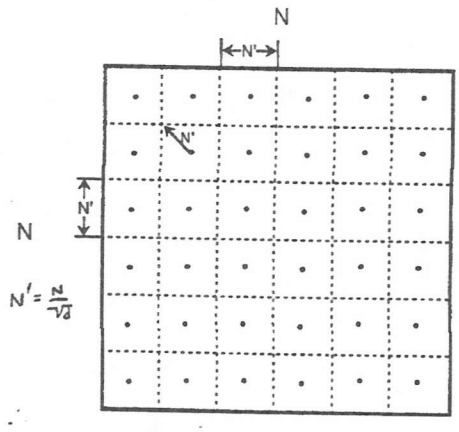

Figure 2. A mesh partitioned into d square segments.

So now we have,

$$\frac{n}{\sqrt{2}\sqrt{d}} \leq delay \leq \frac{n}{\sqrt{d}}.$$

The upper bound above is within a factor of $1/\sqrt{2}$ of the lower bound. Although we are not sure how to make these bounds tighter, we believe that the broadcast algorithm described above is optimal, and that the lower bound is a little loose. Also, if d is not a perfect square we can always use a smaller number d' of start nodes which is a perfect square and is within a small constant factor of d ($d' > d - \sqrt{d} > d/2$). Similarly if n does not divide the square root of d, the segments will not be all of the same size, but there size will still be $< n/\sqrt{d} + 1$.

3 Multiple-node Broadcast Algorithms

We consider an $n \times n$ mesh with n^2 nodes, each of which generates packets independently according to a Poisson random process of rate λ. The packets take exactly one time slot to be transmitted, each node can only transmit to one of his neighbors at a given slot, but can receive from all of his neighbors simultaneously. We begin by providing a necessary condition for stability for any multinode broadcast algorithm.

Proposition 2 *In order for a multinode 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 (this is because $n^2 - 1$ nodes must receive the packet, and no transmission can be heard by more than one node 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, $\lambda n^2(n^2-1) \leq n^2$ must hold, which proves our claim. 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.

3.1 A Simple Synchronized Multiple-node Broadcast Algorithm

The algorithm described in this section is based on performing periodic, complete, multi-node broadcasts, throughout the entire system. A multi-node 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 multi-node broadcast we allow each node to broadcast exactly one packet per cycle. It can be shown that a multi-node broadcast takes $O(n^2 - 1)$ time slots to be performed[3]. Consider the following Multi-node broadcast algorithm which takes $n^2 - 1$ slots. The algorithm proceeds in two steps:

Step 1) Every node broadcasts one message along its vertical ring, so that every node on the ring contains 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 contain all of the messages.

During each slot every node across a vertical ring performs a 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.

3.1.1 Delay Analysis

Clearly the first step 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. Every n^2-1 time units we perform a complete multinode broadcast according to the above algorithm. 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 queues at each node behave as M/D/1 queues with synchronization. That is, we have an M/D/1 queue, where service is offered at prespecified instant 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

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

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.

3.2 A synchronized Multiple-Node broadcast Algorithm

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 ring so that every node is associated with exactly one vertical and one horizontal ring. The algorithm has three stages:

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

Step 2) Every ring selects, at random, up to d packets to be further broadcast through the mesh. The un-selected packets, if any, rejoin their node's queues and attempt retransmission during the next cycle. (If a ring has fewer than d packets then the remaining slots are filled with null packets). All nodes on a given vertical ring choose the same d packets, to be broadcast through the mesh.

step 3) All nodes broadcasting the d "select" packets through their horizontal rings.

As before, the first step takes n-1 slots and the third step takes an additional d(n-1) slots. So, the cycle length, S, is equal to (d+1)(n-1) slots. 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 $\lambda \leq \frac{d}{n(d+1)(n-1)}$. Clearly, for $d < n$ the stability region of this algorithm is smaller than the region implied by 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.

The delay analysis for this scheme is rather complicated. The n queues on each vertical ring are highly dependent on each other. This is because the event of one queue getting service is dependent on whether or not the other queues in the system got served. The queue sizes on 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 computationally prohibitive [4]. Therefore it is useful to develop an approximate model for the analysis of interacting queues. Before we go on to discuss this model, let us first consider a special case of this problem where an exact solution is attainable.

3.2.1 Exact solution for d=1

We can obtain the solution for the case of d equaling one 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 λS (where S = cycle length = (d+1)(n-1)), packets arrive at the ring according to a Poisson process of rate $n\lambda S = 2\lambda n(n-1)$. Therefore the entire ring behaves as an M/D/1 queue with synchronization and with service time $S = 2(n-1)$ and arrival rate $2\lambda n(n-1)$. The average delay for this queue is well known and is given by,

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

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 values of d other than one the entire ring no longer behaves like a 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.

3.2.2 Approximating interacting queues: A two chain approach.

In this approximation we model a n-dimensional infinite Markov chain as a single-dimensional infinite Markov chain together with an n-dimensional finite Markov chain. A similar model was developed previously for the analysis of the Aloha multiple access protocol which also gives rise to interacting queues. We let one chain, the user's chain, represent the queue length for a single node. We let the other chain, the system chain, represent the number of non-empty nodes in the ring. The motivation behind this approach is that the only system information relevant to the analysis of the user chain is the number of nodes in the system attempting transmission. A similar model was developed previously in [2], and later improved in [5], for the analysis of the Aloha multiple access protocol which also gives rise to interacting queues.

The User's Markov Chain

The user's Markov chain contains the queue size for a single user. It is therefore an infinite chain similar to that of a

M/D/1 system with Poisson arrivals. However, unlike the M/D/1 case, departures are not assured and take place with some probability of success P_s. The quantity P_s represents the probability that this node is among the d nodes which are selected for transmission and depends only on the number of nodes in the ring attempting transmission; it can therefore be obtained by conditioning on the state of the system chain. So in fact, what the user chain amounts to is an M/G/1 with geometrically distributed service time.

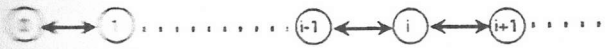

Figure 4. The User Chain.

This, of course, constitutes an approximation to the real model because we assume that the probability of a successful transmission is independent of the number of packets in the queue. The average queue size 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.

$$delay = \frac{S}{2} + \frac{\lambda S(2-\lambda S)}{2\lambda(P_s - \lambda S)},$$

where S is the cycle length and is equal to (d+1)(n-1). 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.

The System Markov Chain

The system chain contains information about the number of non-empty nodes on the ring (not including node x) and the state of node x. Let the pair (M, D_x) represent the state of the system, where M equals the number of non-empty 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. Figure 5 shows the system chain.

Figure 5. The system chain.

Transitions between states require that either some empty nodes receive new packets or that a non-empty node becomes empty after a successful transmission. The latter would not occur unless the non-empty node had exactly one packet in its queue before the transmission took place, and received no new packets. The probability of that event is computed using the statistics of the user chain. Using similar reasoning transition probabilities for this system chain can now be expressed in terms of parameters of the user chain and can be found in [3]. Finally, we need to compute P_s the probability of a successful transmission that is required for the user chain. This probability can be expressed as follows:

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

Where $P_{i,j}$ is the probability of state (i,j). The quantity in the denominator represents the probability that the user chain is not empty. The quantity in the numerator represents the joint probability of a successful transmission and the user chain being not empty. Clearly, when the system has fewer than d non-empty nodes successful transmission takes place with probability one. However, if the system has $i \geq d$ non-empty nodes, successful transmission for the user chain takes place with probability $d/(i+1)$.

We now have the user chain whose solution requires parameters from the system chain (namely, P_s) and the system chain whose solution requires parameters from the user chain. The two systems can be solved together using the Wegstein's iteration scheme[6]. The iterations begin by solving the user chain with $P_s = 1$ and then using the results of the user chain to solve the system chain and obtain a new value for P_s. The iterations continue until the values of the parameters stabilize (the iterations differ by no more than $10^{-6}\%$).

The results of this approximation are compared with simulation for various values of n and d and are presented in [3]. Figure 6 shows this comparison for n=10 and d=4. As can be seen from the figure the approximation performs best when arrival rates are low and similar results are obtained for different values of n and d. This can be because when arrival rates are low very little contention takes place between the nodes for the d available slots and, therefore, there is very little interaction between the queues.

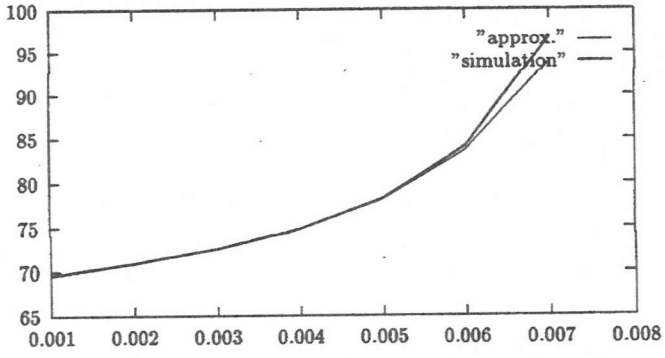

Figure 6. Comparison between approximate results and simulation with n=10 and d=4.

4 Conclusion

This paper presents broadcast schemes for a mesh network with somewhat unusual properties which resulted from a specific application. 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 [2]. 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. Preliminary results show that this new model performs better than existing approximations for the Aloha multiple access protocol[3].

References

[1] G. Stamoulis and J. Tsitsiklis, "Efficient Routing Schemes for Multiple Broadcasts in Hypercubes," *Proc. of the 29th CDC*, Honululu, Hawaii, December 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 Transactions on Automatic Control*, June, 1981.

[3] E. Modiano, "Security and performance issues in distributed protocols," Ph.D. Thesis in preparation, Department of Electrical Engineering, The University of Maryland, College Park, 1991.

[4] T. Nakakis and A. Ephremides, "Steady-State Behavior of Interacting Queues - A Numerical Approach," *IEEE Transactions on Information Theory*, March, 1990.

[5] A. Ephremides and R. Z. Zhu, "Delay Analysis of Interacting Queues with an Approximate Model" *IEEE Transactions on Communications*, Feb., 1987.

[6] G. N. Lance, *Numerical Methods for High Speed Computers*, London: Iliffe, 1960, pp. 134-138.