The Effect of Bandwidth Allocation Policies on Delay in Unidirectional Bus Networks

D. Manjunath, *Member, IEEE,* and Mart L. Molle, *Member, IEEE*

***Abstract*— We consider the problem of allocating bandwidth fairly to each node in a shared, unidirectional bus network. (This bus may represent a complete "folded bus" system, or one side of a "dual bus" network.) If each node simply followed the obvious greedy algorithm, and transmitted its packets in the first available free slot, then the more upstream nodes would enjoy a natural advantage over their downstream neighbors. Thus, a variety of bandwidth allocation policies have been proposed to equalize some measure of the delay, such as the mean access time, or perhaps the mean system time, at different nodes. Several broad categories of access policies have appeared in the literature, including reservation-based schemes (such as DQDB and CRMA), cycle-based schemes (such as Expressnet and Fasnet) and position-dependent flow control schemes (such as the p_i persistent protocol). In this paper, we focus on the last category, since these are open loop policies designed to operate well in high speed networks, which have a very large bandwidth-delay product and feedback in the upstream direction is not available in a timely manner. First, we introduce an improvement to the basic p_i persistent protocol, in which we replace random coin tosses with a deterministic counting algorithm, and thereby reduce the delays for all nodes for any given choice of $\{p_i\}$. We then describe an exact method for calculating average packet delays and queue lengths in both the p_i persistent and our new deterministic n out of m protocols, based on the regenerative approach of Georgiades *et al* [3]. These delay results, together with simulation measurements, show that both of these protocols still waste some bandwidth. After presenting a lower bounding argument to show that some wasted bandwidth is inevitable in *all* such distributed access control schemes, assuming a passive bus without feedback in the upstream direction, we show that changing the bus to unidirectional point-to-point links between (very simple) active interfaces at each node allows us to construct distributed access schemes that require *no* upstream feedback and are *both* work conserving and fair. To illustrate how this can be done, we introduce the p_i preemptive protocol, in which each node randomly inserts its own packets into the traffic arriving from upstream. We derive a simple and effective heuristic for calculating the preemption probability for each node, and use simulation to show how well it equalizes the delays at each node.**

I. Introduction

METROPOLITAN AREA NETWORKS (MAN's) are intended to support high data rate communications over a large geographical area. Thus, shared medium MAN's must employ protocols that can operate under very large bandwidth-delay products (i.e., much larger than the average packet length). One popular topology for these networks is the unidirectional bus, for which a variety of cycle-based, limited service scheduling algorithms have been proposed, such as the Fasnet and Expressnet protocols. Although these cyclic algorithms can equalize the throughput and delay among the nodes (i.e., they are *fair*), they impose a large amount of scheduling overhead because of the gaps between successive cycles, especially as the bandwidth-delay product increases. This overhead reduces throughput and increases delay in comparison to a work conserving scheduling policy on an idealized centralized queuing system.

Manuscript received September 30, 1994; revised April 1, 1995.
D. Manjunath is with the Dept. of Computer Science University of Toronto, Toronto, M5S 1A4 Canada.
M. L. Molle is with the Dept. of Computer Science University of California, Riverside, Riverside, CA 92521 USA.
IEEE Log Number 9413115.

The sensitivity to the bandwidth-delay product can be eliminated by using a distributed bandwidth allocation policy. In a distributed policy, each node along the bus makes a local decision about whether or not to transmit a packet in each of the available slots as they propagate along the bus. Once a node decides to use a given slot, that slot is unavailable to the other nodes located further downstream along the bus, assuming a passive bus architecture. Thus, nodes closer to the head-end of the bus enjoy a natural advantage over their downstream neighbors, as is most evident in the greedy scheduling mechanism where nodes use the first available free for every packet. From the simulation results of [11] we see that this unfair advantage for the upstream can become quite large as the load increases. For example, in a 10-node network where each node generates an average of 0.05 packets per slot, the average access delay for the last node is more than twice as large as for the first node.

In general, distributed policies impose some type of position-dependent flow control scheme on each node, with the goal of reducing the access delays at the downstream hosts at the expense of the upstream hosts. For example, the p_i persistent protocol [12] uses coin tosses to randomly skip some of the available transmission opportunities. Similarly, if we consider the dynamics of the DQDB protocol in the limit of very large propagation delays, we can view the bandwidth balancing (BWB) scheme [4] as a deterministic flow control policy where the node is allowed to use n out of every $n + 1$ transmission opportunities. In this paper, we present some protocol enhancements and new analytical results for a variety of distributed bandwidth allocation policies, including the p_i persistent protocol.

The rest of the paper is organized as follows. First, in Section II we focus on the p_i persistent protocol and show how to improve its delay characteristics by replacing random coin tosses with a new deterministic n out of m algorithm. We then extend the existing analytical model for the p_i persistent protocol, which is based on a simple Bernoulli empty-slot arrival model, to the deterministic algorithm. However, because of the known shortcomings of this model, we use simulation to demonstrate the resulting performance improvement.

Then, in Section III, we develop a new analytical model that yields upper and lower bounds on the average packet delay and queue length in the p_i persistent and the deterministic protocols. This model uses a multi-dimensional version of the regenerative approach to the delay and throughput analysis of random access protocols (see, for example [16], [3]). These bounds are exact and the model allows us to capture the interdependence of the access times for different hosts in the network.

In Section IV we focus on the issue of wasted bandwidth under distributed bandwidth allocation policies. Although the original motivation for development of the p_i persistent protocol was to eliminate the dead time between successive cycles in the cycle-based protocols, we show that considerable bandwidth is wasted in both the p_i persistent and the deterministic protocols. Furthermore, we show that no such protocol for a passive unidirectional bus without feedback can be both fair and work conserving.

Finally, in Section V, we point out that by using a simple active network interface at each node, the downstream nodes can adjust the access delays experienced by their own packets to match the global average by interleaving their own packets into the traffic arriving from upstream. We also describe the p_i *preemptive protocol* in considerable detail, and show that it is both fair and work conserving, and that it is simple to implement and tune. The parameters of this protocol are derived, and an analytical model is presented and the results compared against simulation.

II. Distributed Schemes Under the Approximate Bernoulli Model

A. The p_i Persistent Protocol

In the p_i persistent protocol, empty slots are generated at the head end of a unidirectional bus without feedback. There are N nodes in the network numbered $1, 2, \cdots, N$ starting from the head end of the bus. All packets are of fixed length, and equal to the slot size. Slots are initially marked empty by the head end, and thereafter pass by each node in turn until they are discarded at the tail of the bus. The scheduling rule followed by node i, whenever its queue is non empty, consists of generating a Bernoulli trial with probability p_i whenever an empty slot arrives, and transmitting a packet in that slot if the trial succeeds. Let λ_i be packet arrival rate to node i, and define

$$\Lambda_i = \sum_{j=1}^{i} \lambda_j$$

as the aggregate arrival rate over the first i nodes.

An approximate analysis of the p_i persistent protocol can be found in [12]. Their method assumes that the packet arrivals to each node form an independent Poisson process. In addition, their results also depend on the following Bernoulli approximation, where arrival of empty slots to node i is assumed to be an independent Bernoulli process with mean $(1 - \Lambda_{i-1})$.

Define the access delay for a packet at node i, X_i, as the number of complete slots between its arrival at the head of the queue for node i and the end of its transmission. (Note that we are ignoring the effects of synchronizing each transmission to the ongoing slot boundaries, which can be treated separately.) Under the Bernoulli approximation, X_i has the following geometric distribution

$$\text{Prob}(X_i = k, k > 0) = (1 - (1 - \Lambda_{i-1})p_i)^{k-1}(1 - \Lambda_{i-1})p_i. \tag{1}$$

The first three moments of the access delay at node i, $\overline{x_i}$, $\overline{x_i^2}$ and $\overline{x_i^3}$ respectively are

$$\begin{aligned}\overline{x_i} &= \frac{1}{(1-\Lambda_{i-1})p_i}\\ \overline{x_i^2} &= \frac{2-(1-\Lambda_{i-1})p_i}{((1-\Lambda_{i-1})p_i)^2}\\ \overline{x_i^3} &= \frac{(p_i(1-\Lambda_{i-1}))^2 + 6(1-p_i(1-\Lambda_{i-1}))}{(p_i(1-\Lambda_{i-1}))^3}.\end{aligned} \tag{2}$$

Under the Bernoulli approximation, node i can be treated as an independent M/G/1 queuing system with multiple vacations. However, we assume a slightly different characterization of that queue in comparison to [12]. That is, we define the time from the start of one transmission opportunity for node i until the start of the next to be a *service epoch*. Depending on the queue length at the start of a service epoch, that epoch either becomes a vacation period or the service time for a single customer in the model. Furthermore, we recognize that under the Bernoulli approximation, the sequence of service epoch lengths (and hence vacations and service times) will be i.i.d. random variables. Thus, we can apply some stochastic decomposition results by Doshi [1] and by Levy and Kleinrock [9], to show that the waiting distribution factors into a product form, with the terms representing the residual life of a vacation, the waiting time in an equivalent queue *without* vacations, and the service time, respectively. Let $\overline{x_i}$, $\overline{x_i^2}$ and $\overline{x_i^3}$ be the first three moments of the length of a service epoch, let $\overline{v_i} = \frac{\overline{x_i^2}}{2\overline{x_i}}$ and $\overline{v_i^2} = \frac{\overline{x_i^3}}{3\overline{x_i}}$ be the first two moments of the residual life of vacation, and let

$$\overline{w_i} = \frac{\lambda_i \overline{x_i^2}}{2(1 - \lambda_i \overline{x_i})}$$

and

$$\overline{w_i^2} = 2\overline{w_i} + \frac{\lambda_i \overline{x_i^3}}{3(1 - \lambda_i \overline{x_i})}$$

be the first two moments of the waiting time in an M/G/1 queue. Then we can obtain the average packet delay at node i by inspection, as

$$\overline{d_i} = \overline{w_i} + \overline{v_i} + 1 = \frac{\overline{x_i^2}/\overline{x_i}}{2(1 - \lambda_i \overline{x_i})} + 1. \tag{3}$$

Similarly, because of independence

$$\overline{d_i^2} = \text{var}(v_i) + \text{var}(w_i) + \text{var}(1) + (\overline{v_i} + \overline{w_i} + 1)^2$$

which can be simplified to

$$\overline{d_i^2} = \overline{v_i^2} + \overline{w_i^2} + 2\overline{d_i} + 2\overline{v_i}\,\overline{w_i} - 1. \tag{4}$$

The value of p_i, the probability with which node i transmits its packet in an idle slot, is a parameter of the algorithm, which may be chosen to achieve some fairness criterion. In this case, Mukherjee decided to use p_i to equalize the average delay at all nodes under the Bernoulli free slot approximation. Since the expression for $\overline{d_i}$ depends only on p_i, λ_i, and the combined traffic from the Λ_{i-1}, and since $p_N = 1$ by definition, it is easy to obtain the following expression for p_i [13]

$$p_i = \frac{2(1-\Lambda_N) + \lambda_i(1+\Lambda_{N-1})}{(2-\lambda_N)(1-\Lambda_{i-1})}. \tag{5}$$

Below, we will also need the analogous results when the arrival process at each node is a Bernoulli trial with probability λ_i per slot in discrete time. Since the average waiting time in a discrete time M/G/1 queue is

$$\overline{w_i} = \frac{\lambda_i(\overline{x_i^2} - \overline{x_i})}{2(1-\lambda_i\overline{x_i})}$$

and since the residual life of a vacation is always at least one slot less than the corresponding service epoch (or it would not have been a vacation!), see immediately that, under the discrete time arrival process, (3) becomes

$$\overline{d_i} = \frac{\overline{x_i^2}/\overline{x_i} - 1}{2(1-\lambda_i\overline{x_i})} + 1 \tag{6}$$

and (5) becomes

$$p_i = \frac{1-\Lambda_N + \lambda_i\Lambda_{N-1}}{(1-\lambda_N)(1-\Lambda_{i-1})}. \tag{7}$$

B. A Deterministic Access Scheme

Recall that in an M/G/1 queuing model, such as the one we used to model the p_i persistent scheme under the Bernoulli approximation, waiting times are proportional to the second moment of the service time distribution. Thus, since choosing a more deterministic access control scheme will reduce the second moment of the access delay at each node, we conjecture that it will also reduce the total packet delay and its variance at that node. Furthermore, by smoothing the flow of idle slots to the downstream nodes, their average packet delay should be improved as well. To achieve this, we now consider the following deterministic access rule.

Like the p_i persistent protocol, assume that node i is allowed to use a fraction p_i of the empty slots that arrive from upstream. However, instead of choosing *each free slot* with probability p_i, our deterministic protocol tries to choose *every* $\frac{1}{p_i}$*th free slot* with probability one. Of course if $\frac{1}{p_i}$ is not an integer, then we will vary our selection between the a_ith and $a_i + 1$st slots, where

$$a_i = \lfloor \frac{1}{p_i} \rfloor.$$

Since the overall average must be $\frac{1}{p_i}$, it is easy to see that we should transmit in the a_ith free slot with probability π_i, where

$$a_i\pi_i + (a_i+1)(1-\pi_i) = \frac{1}{p_i}$$

or equivalently

$$\pi_i = 1 + a_i - \frac{1}{p_i}.$$

Note that our inter-attempt time distribution is *not* memoryless (unlike the geometric distribution in the p_i persistent protocol), so the performance of our scheme depends on how the node responds to the event that its queue is empty when a transmission opportunity occurs. Thus, we will assume a "free running" scheduling policy, where the node continues to select free slots for itself at the given rate, whether or not its queue is non empty. This approach seems like the best compromise between freezing the counter at the beginning or end of the next interval, so the given node does not always select the maximum or minimum possible access delay, respectively, for the first packet of its next busy period.

The problem of spreading the empty slots as evenly as possible is equivalent to converting a straight line segment with slope $\frac{1}{p_i}$ into pixels in a computer graphics program. The abscissa of each point represents the sequence number of the transmission opportunity for node i and the corresponding ordinate is the sequence number of the empty slot at node i in which this opportunity is made available. This algorithm may be summarized as follows and is executed at node i.

```
empty_slot_num = 0;
ideal_time = 1 / p_i;
closest_empty_slot = round (ideal_time);
repeat {
  if (arriving_slot is empty) {
    ++empty_slot_num;
    if (empty_slot_num == closest_empty_slot {
      if (transmit_buffer not empty)
        transmit_in_slot;
      ideal_time += 1 / p_i;
      closest_empty_slot = round (ideal_time);
    }
  }
}
```

In general we can assume that p_i is a rational number, say $p_i = \frac{n_i}{m_i}$, in which case an efficient implementation is available using only integer operations known as Bresenham's line drawing algorithm [14]. Thus we shall refer to this algorithm as the deterministic n-out-of-m protocol.

An alternative way of spreading the slots evenly is to use golden ratio weighted TDM policy [6], [5]. In this scheme, p_i is written as $\frac{n_i}{m_i}$ (if p_i is irrational, it is appropriately approximated) where m_i is a number in the Fibonacci sequence. To spread the n_i transmission opportunities evenly among every m_i empty slots that arrive at node i, multiplicative hashing with the golden ratio multiplicand $\phi^{-1} = \frac{\sqrt{5}-1}{2}$ is used. The resulting allotment of slots is very similar to the one obtained using the line drawing algorithm from above.

Given the same assumptions of Poisson packet arrivals and Bernoulli approximation for the empty slot arrivals, we can easily adapt the analysis in [12] to the deterministic n-out-of-m protocol. The only change required is to the access delay at node i, X_i, which now requires us to wait for the a_ith or $a_i + 1$st free slot, with respective probabilities π_i and $1 - \pi_i$.

Since the event that $X_i = k$, given the algorithm must select the a_ith free slot, is equivalent to the event that exactly $a_i - 1$ of the first $k - 1$ arriving slots were busy and the kth arriving slot was idle, we have

$$\text{Prob}(X_i = k|a_i) = \binom{k-1}{a_i - 1}(\Lambda_{i-1})^{k-a_i}(1 - \Lambda_{i-1})^{a_i} \quad \text{for } k \geq a_i. \tag{8}$$

We can similarly obtain the access delay distribution for the packets that must wait for the a_i + 1st empty slot. From this we calculate the first and the second moments of the access delay as

$$\overline{x_i} = \frac{1}{(1 - \Lambda_{i-1})p_i}$$
$$\overline{x_i^2} = \frac{1 + \Lambda_{i-1} + a_i(2 - p_i - a_i p_i)}{p_i(1 - \Lambda_{i-1})^2}. \tag{9}$$

The third moment of the access delay can also be calculated using (8). As in the case of the p_i persistent protocol, we can then obtain the first and second moments of the packet delay for the deterministic protocol using the M/G/1 vacation model.

Comparing (2) and (9) we note that for a given load profile (Λ_{i-1} and λ_i) and p_i, the mean access delay in the p_i persistent and the deterministic algorithm are identical. However, we can show that the second moment of the access delay for the p_i persistent scheme is greater than that for the deterministic scheme. From (2) and (9) this is true if

$$\frac{2 - (1 - \Lambda_{i-1})p_i}{p_i} \geq 1 + \Lambda_{i-1} + a_i(2 - p_i - a_i p_i)$$

or equivalently

$$2 > (1 - b_i^2)(1 + \frac{1}{a_i})$$

where $b_i = 1 - a_i p_i$. Since $a_i = \lfloor \frac{1}{p_i} \rfloor$ and $0 < p_i \leq 1.0$, $0.0 \leq b_i < 1.0$. Hence $(1 - b_i^2) < 1$ and $(1 + \frac{1}{a_i}) < 2$. Hence, the last inequality is satisfied. Thus, according to the Bernoulli approximation method for the empty slot arrival process, the deterministic protocol will have a lower total packet delay than the p_i persistent protocol.

C. Comparison with Simulation

We now compare the accuracy of the Bernoulli approximation with simulation results for both the p_i persistent and n-out-of-m protocol. In Figs. 1 and 2 we plot the first and the second moments of the total packet delay at the nodes in a 10-node unidirectional bus, respectively. We used three values of Λ_{10}, the total packet arrival rate to the network. The results shown are for uniform loading, in which the packet arrival rate is the same for all nodes. Similar results were found when we considered proportional loading, where the packet arrival rate to a node decreases linearly as we move downstream on the bus.

As can be seen in Fig. 1 the Bernoulli approximation model works well for the first half of the bus, but it becomes grossly optimistic as we move further downstream, especially for moderate to high loads. Because of this loss of modeling accuracy as we move to the downstream nodes, the calculated values of p_i do not achieve the intended level of fairness. Indeed, performance degrades significantly as we move downstream along the bus. This unfairness effect was even larger in the proportionally loaded network. We also see that for the same set $\{p_i\}$, the deterministic n-out-of-m protocol is uniformly better than the p_i persistent protocol. However, since the upstream nodes enjoy most of the improvement, fairness would be degraded by this change unless new p_i values were selected.

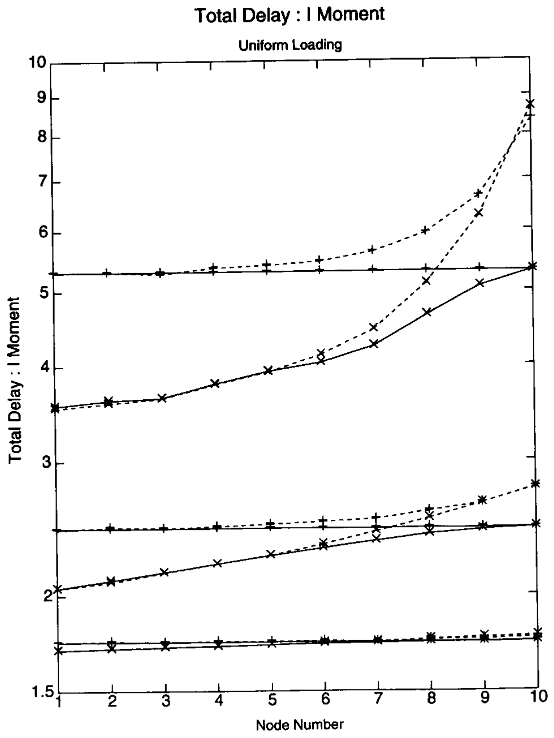


Fig. 1. Total packet delay for the p_i persistent (indicated by "+") and the deterministic n-out-of-m (indicated by "x") protocols at λ = 0.2, 0.5, and 0.8 via analysis (solid lines) and simulation (dashed lines).

Fig. 2 shows that the Bernoulli approximation model is less successful at predicting the second moment of the total packet delay than its mean. Furthermore, the second moment is even more sensitive to position along than bus than the mean.

III. Exact Bounds on Queue Lengths and Waiting Times

In the previous section, we showed that the Bernoulli approximation model breaks down for the downstream nodes under all loads, and for the remaining nodes under moderate to high loads. In this section we present a method for obtaining upper and lower bounds on the average delay and queue lengths at a node in the p_i persistent and the deterministic protocols. We first present a detailed discussion of the delay analysis of the p_i persistent protocol in Section II-A. The model to obtain average queue lengths in the p_i persistent protocol is then presented in Section III-B. A brief overview of the technique as adapted to the deterministic protocol

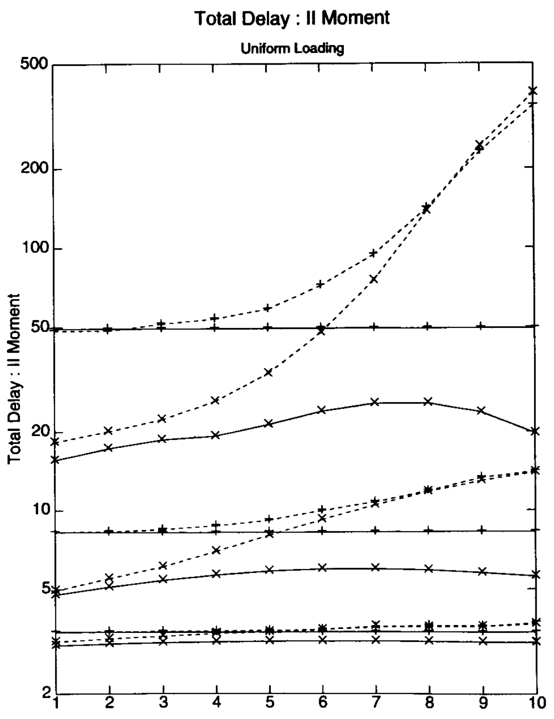


Fig. 2. Second moment of total packet delay for the p_i persistent (indicated by "+") and the deterministic n-out-of-m (indicated by "×") protocols at $\lambda = 0.2$, 0.5, and 0.8 via analysis (solid lines) and simulation (dashed lines).

is presented in Section III-C. Numerical results from the analysis and comparison with simulation results and Bernoulli Approximations is presented in Section III-D.

A. Delays in the p_i Persistent Protocol

Consider a network with N nodes. Assume that the packet arrival process to node i is a Bernoulli process in discrete time, with mean λ_i per slot, and is independent of the arrivals to the other nodes. (The advantage of adopting this discrete time arrival process, instead of a Poisson process, is that it gives us a much simpler model, since the access delays are restricted to a countable state space.) Let the packet lengths be equal to one slot length. Also, assume that the arrivals occur at the beginning of a slot and that the decision to transmit is made just after the arrival instants. The bus is unidirectional and there is no feedback from a node to nodes upstream from it. Furthermore, the packet arrival process in a slot at a node is memoryless, time homogeneous, and independent of the packet arrival process at the other nodes. Hence, without loss of generality, we can ignore the propagation delays between each successive node pair by appropriately time-shifting the arrival processes at each node.

Define the lag at node i during slot k, $l_i(k)$, to be the delay that the packet (if any) at the head of the queue would suffer if it were transmitted in slot k. If there are no packets in a node's packet queue at the beginning of the slot, the lag is defined to be 0. An arriving packet brings in a lag of 1. Define an

TABLE I
CURRENT STATE, POSSIBLE NEXT STATES AND PROBABILITIES FOR NODE i. p_i PERSISTENT PROTOCOL WITH BERNOULLI ARRIVALS

Current Values		Event		New Values	
$U_i(k)$	$l_i(k)$	Type	Probability	$U_{i+1}(k)$	$l_i(k+1)$
Busy	>0	cant Tx	1	Busy	$l_i(k)+1$
Busy	0	arvl in slot $k+1$	λ_i	Busy	1
Busy	0	No arvl in slot $k+1$	$1-\lambda_i$	Busy	0
Idle	>0	Tx	$p_i f_i(j_i)$	Busy	$l_i(k)-j_i$
Idle	>0	dont Tx	$1-p_i$	Idle	$l_i(k)+1$
Idle	0	arvl in slot $k+1$	λ_i	Idle	1
Idle	0	no arvl in slot $k+1$	$1-\lambda_i$	Idle	0

$$f_i(j_i) = \begin{cases} \lambda_i(1-\lambda_i)^{j_i} & \text{for } j_i = 0, 1, \cdots, l_i(k)-1 \\ (1-\lambda_i)^{l_i(k)} & \text{for } j_i = l_i(k) \end{cases}$$

N-dimensional vector $\mathbf{l}(k) = [l_1(k), \cdots, l_N(k)]$. (Note that in the case of nonzero propagation delays, these components should be evaluated when the given slot reaches each node, rather than at some absolute time instant. Let $U_i(k)$ be the status of the k^{th} slot (busy or idle) that arrives at node i. For a given lag at node i and the state of the slot when it arrives at node i, we list the possible events at node i, the lag at the node at the beginning of the next slot arrives and the probability of the event in Table I. Every time node i transmits, the reduction in the lag is geometrically distributed because of the Bernoulli arrival process.

Embed a Markov process at the beginning of the slot just after the arrival instants and use $\mathbf{l}(k)$ as the state description for the system in slot k. The transition probability matrix for this system can then be constructed using Table I. In this system, the state $\mathbf{l} = \mathbf{0} = [0, 0, \cdots 0]$ is seen to be a regeneration point in the evolution of $\mathbf{l}(k)$. The number of slots between successive regeneration points is the length of the *regenerative cycle*. Let $C(\mathbf{l})$ be the average number of slots needed to go from state $\mathbf{l}$ to the next regeneration point, state $\mathbf{0}$. The operation of the protocol gives us the following relation for $C(\mathbf{l})$.

$$C(\mathbf{l}) = \mathbf{1} + \sum_{\mathbf{l}' \neq \mathbf{0}} \mathbf{C}(\mathbf{l}')\, \mathbf{P}_{\mathbf{l},\mathbf{l}'} \tag{10}$$

where $\mathrm{P}_{\mathbf{l},\mathbf{l}'}$ is the one step transition probability from state $\mathbf{l}$ to state $\mathbf{l}'$.

From the definition of the lag, if node i transmits when its lag is l_i, the total delay of the packet that was transmitted will be l_i. Denote by $D_i(\mathbf{l})$ the packet delays accumulated by the packets that node i transmits as the system evolves from state $\mathbf{l}$ to state $\mathbf{0}$. Clearly, $D_i(\mathbf{l})$ is incremented only if there is a transmission from node i. Let $\theta_i(\mathbf{l})$ be the probability of node i transmitting when the system is in state $\mathbf{l}$. We can write the equation for $D_i(\mathbf{l})$ as follows

$$D_i(\mathbf{l}) = l_i\theta_i(\mathbf{l}) + \sum_{\mathbf{l}' \neq \mathbf{0}} D_i(\mathbf{l}')\, \mathrm{P}_{\mathbf{l},\mathbf{l}'} \tag{11}$$

where l_i is the i^{th} element of $\mathbf{l}$.

The average cycle length, $\overline{C}$, is $C(\mathbf{0})$ and the average cumulative packet delay at node i during a cycle, $\overline{D_i}$, is

TABLE II
TRANSITION PROBABILITIES TO CALCULATE THE AVERAGE PACKET DELAY BOUNDS IN THE TWO NODE p_i PERSISTENT PROTOCOL NETWORK

cur state$[l_1, l_2]$	event	next state$[l'_1, l'_2]$	probability
$[0,0]$	no arvls to 1 & 2	$[0,0]$	$(1-\lambda_1)(1-\lambda_2)$
	no arvl to 1, arvl to 2	$[0,1]$	$(1-\lambda_1)\lambda_2$
	arvl to 1, no arvl to 2	$[1,0]$	$\lambda_1(1-\lambda_2)$
	arvls to 1 & 2	$[1,0]$	$\lambda_1\lambda_2$
$[0, l_2 > 0]$	no arvl to 1	$[0, l_2 - j_2]$	$(1-\lambda_1)f_2(j_2)$
	arvl to 1	$[1, l_2 - j_2]$	$\lambda_1 f_2(j_2)$
$[l_1 > 0, 0]$	1 doesnt Tx, no arvl to 2	$[l_1+1, 0]$	$(1-p_1)(1-\lambda_2)$
	1 doesnt Tx, arvl to 2	$[l_1+1, 1]$	$(1-p_1)\lambda_2$
	1 Txs, no arvl to 2	$[l_1 - j_1, 0]$	$p_1 f_1(j_1)(1-\lambda_2)$
	1 Txs, arvl to 2	$[l_1 - j_1, 1]$	$p_1 f_1(j_1)\lambda_2$
$[l_1 > 0, l_2 > 0]$	1 Txs, 2 cant Tx	$[l_1 - j_1, l_2 + 1]$	$p_1 f_1(j_1)$
	1 doesnt Tx, 2 Txs	$[l_1 + 1, l_2 - j_2]$	$(1-p_1)f_2(j_2)$

$$f_i(j_i) = \begin{cases} \lambda_i(1-\lambda_i)^{j_i} & \text{for } j_i = 0, 1, \cdots, l_i - 1 \\ (1-\lambda_i)^{l_i} & \text{for } j_i = l_i \end{cases}$$

$D_i(\mathbf{0})$. From Wald's theorem, the average number of packets transmitted by node i during a cycle is $\lambda_i \overline{C}$. Using the theory of regenerative processes, the average delay of a packet transmitted at node i, $\overline{d_i}$, is given by

$$\overline{d_i} = \frac{\overline{D_i}}{\lambda_i \overline{C}}$$

For an N-node system, $\mathbf{l} \in \mathcal{Z}_+^N$, where $\mathcal{Z}_+$ is the set of non negative integers. Clearly, it is not possible to solve (10) and (11) to obtain $\overline{C}$ and $\overline{D_i}$ because the system is a set of infinite equations. However, upper and lower bounds for $\overline{C}$ and $\overline{D_i}$ can be found using the techniques used in the analysis of random access systems, (see for example [16], [2], [3]), wherein a system of infinite equations is bounded by a homogeneous solution subject to finite number of boundary equations and solved using the techniques outlined in [7]. The upper and lower bounds on the average delays at node i, $\overline{d_i}^{(u)}$ and $\overline{d_i}^{(l)}$ respectively, are then given by

$$\overline{d_i}^{(u)} = \frac{\overline{D_i}^{(u)}}{\lambda_i \overline{C}^{(l)}} \qquad \overline{d_i}^{(l)} = \frac{\overline{D_i}^{(l)}}{\lambda_i \overline{C}^{(u)}}$$

where $\overline{D_i}^{(u)}$ and $\overline{C}^{(u)}$ are the upper bounds on $\overline{D_i}$ and $\overline{C}$ respectively and $\overline{D_i}^{(l)}$ and $\overline{C}^{(l)}$ are the corresponding lower bounds.

We now restrict ourselves to the case of $N = 2$. For a given state, in Table II we list the possible events, the corresponding next state and the probabilities of these events in that state for a two node system.

The Bernoulli analysis of Section II-A is exact for the first node. Hence, we focus on node 2.

First, truncate the system of infinite equations of (10) and (11) for a two node network to a finite system and consider $\mathbf{l} \in \mathcal{A}$, where $\mathcal{A} \subset \mathcal{Z}^2$. If we assume

$$C(\mathbf{l}) = \alpha_1 l_1 + \alpha_2 l_2 + \beta.$$

Rewriting (10), for the truncated system over $\mathcal{A}$, we get

$$\begin{aligned} C(\mathbf{l}) &= 1 + \sum_{\mathbf{l}' \neq \mathbf{0},\, \mathbf{l}' \in \mathcal{A}} C(\mathbf{l}')\mathrm{P}_{\mathbf{l},\mathbf{l}'} + \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} C(\mathbf{l}')\mathrm{P}_{\mathbf{l},\mathbf{l}'} \\ &= 1 + \sum_{\mathbf{l}' \neq \mathbf{0},\, \mathbf{l}' \in \mathcal{A}} C(\mathbf{l}')\mathrm{P}_{\mathbf{l},\mathbf{l}'} + \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} (\alpha_1 l'_1 + \alpha_2 l'_2 + \beta)\mathrm{P}_{\mathbf{l},\mathbf{l}'} \\ &= 1 + \sum_{\mathbf{l}' \neq \mathbf{0}} C(\mathbf{l}')\mathrm{P}_{\mathbf{l},\mathbf{l}'} + \psi(\mathbf{l}) \qquad \text{for } \mathbf{l} \in \mathcal{A} \qquad (12) \end{aligned}$$

where $\psi(\mathbf{l}) = \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} (\alpha_1 l'_1 + \alpha_2 l'_2 + \beta)\mathrm{P}_{\mathbf{l},\mathbf{l}'} = \alpha_1 \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} l'_1 \mathrm{P}_{\mathbf{l},\mathbf{l}'} + \alpha_2 \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} l'_2 \mathrm{P}_{\mathbf{l},\mathbf{l}'} + \beta \sum_{\mathbf{l}' \in \overline{\mathcal{A}}} \mathrm{P}_{\mathbf{l},\mathbf{l}'} = \alpha_1 \psi_1(\mathbf{l}) + \alpha_2 \psi_2(\mathbf{l}) + \beta \psi_0(\mathbf{l})$.

Choosing α_1, α_2 and β such that

$$C(\mathbf{l}) \geq \alpha_1 l_1 + \alpha_2 l_2 + \beta \qquad \forall\ \mathbf{l}$$

and iterating on (12), we can obtain a lower bound on $C(\mathbf{l})$ and hence a lower bound on $\overline{C}$. When the system is in state $\mathbf{l} = [l_1, l_2]$, the average number of packets that could be in the system is $l_1\lambda_1 + l_2\lambda_2$. Assuming that the access delay for these packets is exactly 1 slot and no other packets arrive during their service, a lower bound on the time till the next regeneration point is $l_1\lambda_1 + l_2\lambda_2$. Therefore, we choose $\alpha_1 = \lambda_1$, $\alpha_2 = \lambda_2$ and $\beta = 1$, start with an initial value of $C(\mathbf{l}) = \alpha_1 l_1 + \alpha_2 l_2 + \beta$ for all $\mathbf{l} \in \mathcal{A}$ and iterate on (12) to obtain $\overline{C}^{(l)}$.

Similarly, choosing α_1, α_2 and β such that

$$C(\mathbf{l}) \leq \alpha_1 l_1 + \alpha_2 l_2 + \beta \qquad \forall\ \mathbf{l}$$

and iterating on (12), we can obtain an upper bound on $\overline{C}$. For bounded delays, $\lambda_1 < p_1$. Consider an auxiliary discrete time queuing system that has Bernoulli arrivals with a mean of λ_2 arrivals per slot and geometric service times with mean $\frac{1}{1-p_1}$ and a setup time of $l_1 + l_2$. The average busy period durations in this auxiliary queuing system will be greater than the average number of slots from state $[l_1, l_2]$ to the next regeneration point in the two node p_i persistent protocol network we are considering. Hence, the average length of the busy period in the auxiliary system will be an upper bound on $C(\mathbf{l})$ where $\mathbf{l} = [l_1, l_2]$. The average busy period duration of the auxiliary system may be obtained by working out the generalized busy period analysis from [8] for a discrete time system with Bernoulli arrivals, geometric service distributions and a setup time of $l_1 + l_2$ at the beginning of the busy period. This is found to be $l_1 + l_2 + [(l_1 + l_2)\lambda_2 + 1]\left[\frac{1}{1-\lambda_2-p_1}\right]$. We therefore choose, $\alpha_1 = \alpha_2 = \frac{1-p_1-p_1\lambda_2}{1-\lambda_2-p_1}$ and $\beta = \frac{1}{1-\lambda_2-p_1}$, choose the initial values of $C(\mathbf{l}) = \alpha_1 l_1 + \alpha_2 l_2 + \beta$ for $\mathbf{l} \in \mathcal{A}$ and iterate on (12) to obtain $\overline{C}^{(u)}$.

For the auxiliary queuing system considered above to be a stable queuing system, and hence for the above model to be valid, we need to have $\lambda_2 + p_1 < 1.0$. For p_1 calculated according to (5), it can be shown that $\lambda_2 + p_1 < 1.0$ is true if and only if $\lambda_2 < \lambda_1$.

To obtain bounds on $\overline{D_2}$, we first assume $D_2(\mathbf{l}) = \gamma_1 l_1^2 + \gamma_2 l_2^2 + \delta$. Along the lines of the derivation of (12), rewriting (11), for a two node network we get,

$$D_2(\mathbf{l}) = l_2\theta_2(\mathbf{l}) + \sum_{\mathbf{l}'\neq\mathbf{0}} C(\mathbf{l}')\mathrm{P}_{\mathbf{l},\mathbf{l}'} + \phi(\mathbf{l}) \qquad \text{for } \mathbf{l} \in \mathcal{A} \quad (13)$$

where $\phi(\mathbf{l}) = \sum_{\mathbf{l}'\in\overline{\mathcal{A}}} (\gamma_1 l_1'^2 + \gamma_2 l_2'^2 + \delta)\mathrm{P}_{\mathbf{l},\mathbf{l}'} = \gamma_1 \sum_{\mathbf{l}'\in\overline{\mathcal{A}}} l_1'^2 \mathrm{P}_{\mathbf{l},\mathbf{l}'} + \gamma_2 \sum_{\mathbf{l}'\in\overline{\mathcal{A}}} l_2'^2 \mathrm{P}_{\mathbf{l},\mathbf{l}'} + \delta \sum_{\mathbf{l}'\in\overline{\mathcal{A}}} \mathrm{P}_{\mathbf{l},\mathbf{l}'} = \gamma_1\phi_1(\mathbf{l}) + \gamma_2\phi_2(\mathbf{l}) + \delta\phi_0(\mathbf{l})$.

When the system is in state l, for each potential arrival time j, $0 < j < l2$, there may be a packet, independently, with probability λ_2. Since the earliest that any of these packets could depart is in the current slot, and expectation is a linear operator, a lower bound on the average cumulative waiting time from the current slot to the next regeneration point is $l_2 + \sum_{j=1}^{l_2-1} j\lambda_2 = l_2(1 + \frac{\lambda_2(l_2-1)}{2})$. Hence, choose $\gamma_1 = 0$, $\gamma_2 = \frac{\lambda_2}{2}$ and $\delta = 0$. $\overline{D_2}^{(l)}$ is obtained by starting with $D_2(\mathbf{l}) = \gamma_1 l_1 + \gamma_2 l_2 + \delta$ for $\mathbf{l} \in \mathcal{A}$ and iterating on (13).

To obtain an upper bound on the average cumulative waiting time from state l to the next regeneration point, we consider our auxiliary queuing system again. Consider a busy period of length of X. The average of the number of customers served in this busy period is $\lambda_2 X$. If we assume that all these customers arrived at the beginning of the busy period and departed at the end of the busy period, we can assign a delay of X to each of the customers served in the busy period. Hence, for a busy period of duration X, $\lambda_2 X^2$ is an upper bound on the cumulative delay for the busy period. Taking expectations, we see that $\lambda_2\overline{X^2}$ is an upper bound on the average cumulative waiting time during a busy period. For an upper bound on $D_2(\mathbf{l})$, we obtain the second moment of the busy period in the auxiliary queuing system that we considered above which is a second degree polynomial in l_1 and l_2. Since we want an upper bound, we obtain an equation of the form $\gamma_1 l_1^2 + \gamma_2 l_2^2 + \delta$ that bounds the polynomial from above and obtain $\gamma_1 = \gamma_2 = \lambda_2 \frac{3(1-p_1)^3 + 2(1-p_1)^2 - 4\lambda_2(1-p_1)^2 - \lambda_2^2(1-p_1)}{(1-p_1-\lambda_2)^3}$ and $\delta = \lambda_2 \frac{2(1-p_1) - (1-p_1)^2 - \lambda_2^2}{(1-p_1-\lambda_2)^3}$. We then start with an initial value of $D_2(\mathbf{l}) = \gamma_1 l_1^2 + \gamma_2 l_2^2 + \delta$ for $\mathbf{l} \in \mathcal{A}$ and iterate on (13) to obtain $\overline{D_2}^{(u)}$.

B. Queue Lengths in the p_i Persistent Protocol

To obtain the average queue length as observed at the beginning of a slot, we proceed as with the evaluation of the average waiting time. However, since the state space must encode the queue size at each node instead of the lag, and queue size is already discrete, there is no need to restrict our analysis to the discrete time Bernoulli per slot arrival process. Let $f_i(j_i)$ be the probability of there being j_i arrivals to node i in a slot. Let $q_i(k)$ be the queue length at node i at the beginning of slot k and $\mathbf{q}(k) = [q_1(k), \cdots q_N(k)]$. As before, $\mathbf{q} = \mathbf{0}$ can be seen to be a regeneration point in the evolution of $\mathbf{q}$. Let $C(\mathbf{q})$ be the average number of slots to go from state $\mathbf{q}$ to state $\mathbf{0}$. From the operation of the protocol, we have

$$C(\mathbf{q}) = 1 + \sum_{\mathbf{q}'\neq\mathbf{0}} C(\mathbf{q}')P_{\mathbf{q},\mathbf{q}'}. \qquad (14)$$

Let $Q_i(\mathbf{q})$ be the average of the cumulative queue lengths at node i at the slot beginnings as the system evolves from state $\mathbf{q}$ to state $\mathbf{0}$. We can write

$$Q_i(\mathbf{q}) = q_i + \sum_{\mathbf{q}'\neq\mathbf{0}} Q(\mathbf{q}')P_{\mathbf{q},\mathbf{q}'}. \qquad (15)$$

The average queue length at node i, $\overline{q_i}$, is calculated using the relation

$$\overline{q_i} = \frac{\overline{Q_i}}{\overline{C}}$$

where $\overline{Q_i} = Q_i(\mathbf{0})$ and $\overline{C} = C(\mathbf{0})$ are the average cumulative queue length over a cycle and the average cycle length respectively.

We now consider a two node network. For a two node network, for various $\mathbf{q}$, we have listed the possible events, the resulting next state and the probabilities in Table III.

To obtain the bounds we assume, $C(\mathbf{q}) = \alpha_1 q_1 + \alpha_2 q_2 + \beta$ and truncate the infinite system of equations of (14) to consider only $\mathbf{q} \in \mathcal{A}$. Rewriting and simplifying we get

$$C(\mathbf{q}) = 1 + \sum_{\mathbf{q}'\neq\mathbf{0}} C(\mathbf{q}')\mathrm{P}_{\mathbf{q},\mathbf{q}'} + \psi(\mathbf{q}) \qquad \text{for } \mathbf{q} \in \mathcal{A} \quad (16)$$

where $\psi(\mathbf{q})$ is similar to the definition of $\psi(\mathbf{l})$ in (12). To obtain a lower bound, assuming no more arrivals till the next regeneration point, we choose $\alpha_1 = \alpha_2 = 1$ and $\delta = 0$. Starting with $C(\mathbf{q}) = \alpha_1 q_1 + \alpha_2 q_1 + \beta$ for $\mathbf{q} \in \mathcal{A}$, we iterate on (16) and obtain a lower bound on the cycle length. To obtain an upper bound, we consider the auxiliary system of a discrete time queue with the number of arrivals in a slot having distribution $f_2(j_2)$ and the service time having a geometric distribution with mean $\frac{1}{1-p_1}$. We also assume that there are $q_1 + q_2$ customers in the system at the start of the busy period. From the busy analysis of this auxiliary system we obtain $\alpha_1 = \alpha_2 = \beta = \frac{\lambda_2}{(1-p_1-\lambda_2)(1-e^{-\lambda_2})}$.

For the bounds on the average cumulative queue length in a regenerative cycle, we assume $Q_2(\mathbf{q}) = \gamma_1 q_1^2 + \gamma_2 q_2^2 + \delta$ and write the truncated system as

$$Q_2(\mathbf{l}) = q_2 + \sum_{\mathbf{q}'\neq\mathbf{0}} C(\mathbf{q}')\mathrm{P}_{\mathbf{q},\mathbf{q}'} + \phi(\mathbf{q}) \qquad \text{for } \mathbf{q} \in \mathcal{A} \quad (17)$$

TABLE III
TRANSITION PROBABILITIES TO CALCULATE THE AVERAGE QUEUE LENGTH BOUNDS IN THE TWO-NODE p_i PERSISTENT PROTOCOL NETWORK

cur state $[q_1, q_2]$	**event**	**next state** $[q'_1, q'_2]$	**probability**
$[0,0]$	nobody Txs	$[j_1, j_2]$	$f_1(j_1)f_2(j_2)$
$[0, q_2 > 0]$	2 Txs	$[j_1, q_2 + j_2 - 1]$	$f_1(j_1)f_2(j_2)$
$[q_1 > 0, 0]$	1 Txs, 0 arvls to 1	$[q_1 - 1, j_2]$	$p_1 f_1(0) f_2(j_2)$
	1 Txs, $j_1 + 1$, $(j_1 > 0)$ arvls to 1 **OR** 1 doesnt Tx, j_1 arvls to 1	$[q_1 + j_1, j_2]$	$p_1 f_1(j_1+1)f_2(j_2)+$ $(1-p_1)f_1(j_1)f_2(j_2)$
$[q_1 > 0, q_2 > 0]$	1 Txs, 0 arvls to 1	$[q_1 - 1, q_2 + j_2]$	$p_1 f_1(0) f_2(j_2)$
	2 Txs, 0 arvls to 2	$[q_1 + j_1, q_2 - 1]$	$(1-p_1)f_1(j_1)f_2(0)$
	1 Txs, $j_1 + 1$ arvls to 1, j_2 to 2 **OR** 2 Txs, j_1 arvls to 1, $j_2 + 1$ to 2	$[q_1 + j_1, q_2 + j_2]$ $j_1, j_2 \geq 0$	$p_1 f_1(j_1+1)f_2(j_2)+$ $(1-p_1)f_1(j_1)f_2(j_2+1)$

$f_i(j_i) = \text{Prob}(j_i \text{ arrivals at node } i \text{ in a slot}) \quad j_i = 0, 1, 2, \cdots$

TABLE IV
TRANSITION PROBABILITIES TO CALCULATE THE AVERAGE QUEUE LENGTH BOUNDS IN THE TWO-NODE DETERMINISTIC n-OF-m PROTOCOL NETWORK WITH BERNOULLI ARRIVALS

cur state$[l_1, y_1, l_2]$	**event**	**next state** $[l'_1, y'_1, l'_2]$	**probability**
$[0, y_1, 0]$	no arvls to 1 & 2	$[0, y_1 + 1, 0]$	$(1-\lambda_1)(1-\lambda_2)$
	no arvl to 1, arvl to 2	$[0, y_1 + 1, 1]$	$(1-\lambda_1)\lambda_2$
	arvl to 1, no arvl to 2	$[1, y_1 + 1, 0]$	$\lambda_1(1-\lambda_2)$
	arvls to 1 & 2	$[1, y_1 + 1, 1]$	$\lambda_1\lambda_2)$
$[0, y_1, l_2 > 0]$	arvl to 1	$[1, y_1 + 1, l_2 - j_2]$	$\lambda_1 f_2(j_2)$
	no arvl to 1	$[0, y_1 + 1, l_2 - j_2]$	$(1-\lambda_1)f_2(j_2)$
$[l_1 > 0, y_1 \notin \mathcal{Y}_1, 0]$	1 cant Tx, no arvl to 2	$[l_1 + 1, y_1 + 1, 0]$	$1 - \lambda_2$
	1 cant Tx, arvl to 2	$[l_1 + 1, y_1 + 1, 1]$	λ_2
$[l_1 > 0, y_1 \in \mathcal{Y}_1, 0]$	1 Txs, no arvl to 2	$[l_1 - j_1, y_1 + 1, 0]$	$f_1(j_1)(1-\lambda_2)$
	1 Txs, arvl to 2	$[l_1 - j_1, y_1 + 1, 1]$	$f_1(j_1)\lambda_2$
$[l_1 > 0, y_1 \notin \mathcal{Y}_1, l_2 > 0]$	1 cant Tx, 2 Txs	$[l_1 + 1, y_1 + 1, l_2 - j_2]$	$f_2(j_2)$
$[l_1 > 0, y_1 \in \mathcal{Y}_1, l_2 > 0]$	1 Txs, 2 cant Tx	$[l_1 - j_1, y_1 + 1, l_2 + 1]$	$f_1(j_1)$

$$f_i(j_i) = \begin{cases} \lambda_i(1-\lambda_i)^{j_i} & \text{for } j_i = 0, 1, \cdots, l_i(k) - 1 \\ (1-\lambda_i)^{l_i(k)} & \text{for } j_i = l_i(k) \end{cases}$$

All increments of y_1 are modulo m.

where $\phi(\mathbf{q})$ is defined like $\phi(\mathbf{q})$ in (13). To obtain the lower bound, assuming no further arrivals till the next regenerative point, we get $\gamma_1 = \delta = 0$, $\gamma_2 = 0.5$. The average cumulative queue length will be bounded above by $\lambda_2 \overline{X^2}$ where $\overline{X^2}$ is the second moment of the busy period of the auxiliary queuing system. From this we obtain $\gamma_1 = \gamma_2 = \lambda_2 \frac{3(1-p_1)^2}{(1-p_1-\lambda_2)^2}$ and $\delta = \lambda_2 \frac{1-p_1-\lambda_2 p_1^2}{(1-p_1)^3}$.

C. The Deterministic Protocol

Obtaining the delay and average queue length bounds for the deterministic n-of-m protocol introduced in Section II-B is along the same lines as that of the p_i persistent protocol, except that the system description needs to include the status of the idle slot counters, y_i at node i.

Consider the waiting time analysis in a two node network. Let $l_i(k)$ be the lag at node i in slot k. Node 2 will use all the free slots that it sees when its queue is not empty. Node 1 keeps track of the idle slot sequence number that it sees in the variable y_1. The vector $\mathbf{l} = [l_1, y_1, l_2]$ describes the state of the network. Let $\mathcal{Y}_1$ be the set of y_1 for which node 1 can transmit in the next empty slot. The set $\mathcal{Y}_1$ can be determined from the protocol description of section 2.2.

From the operation of the algorithm, for various $\mathbf{l}$, we have listed the possible events, the resulting next state and the probabilities in Table IV.

To obtain a regeneration point in the evolution of the state of the system, we have to assume that m and n are integers and the counter y_1 which keeps track of the number of idle slots that arrive at node 1, is a modulo m counter. With this assumption the regeneration point is the state $\mathbf{0} = [0, 0, 0]$. The equations to obtain the average cycle length and the cumulative waiting times are similar to (12) and (13) in the delay analysis

TABLE V
TRANSITION PROBABILITIES TO CALCULATE THE AVERAGE QUEUE LENGTH BOUNDS IN THE TWO-NODE DETERMINISTIC n-OF-m PROTOCOL

cur state $[q_1, y_1, q_2]$	**event**	**next state** $[q_1', y_1', q_2']$	**probability**
$[0, y_1, 0]$	nobody Txs	$[j_1, y_1, j_2]$	$f_1(j_1)f_2(j_2)$
$[0, y_1, q_2 > 0]$	2 Txs	$[j_1, y_1, q_2 + j_2 - 1]$	$f_1(j_1)f_2(j_2)$
$[q_1 > 0, y_1 \notin \mathcal{Y}_1, 0]$	1 cant Tx	$[q_1 + j_1, y_1 + 1, j_2]$	$f_1(j_1)f_2(j_2)$
$[q_1 > 0, y_1 \in \mathcal{Y}_1, 0]$	1 Txs	$[q_1 + j_1 - 1, y_1 + 1, j_2]$	$f_1(j_1)f_2(j_2)$
$[q_1 > 0, y_1 \notin \mathcal{Y}_1, q_2 > 0]$	1 cant Tx, 2 Txs	$[q_1 + j_1, y_1 + 1, q_2 + j_2 - 1]$	$f_1(j_1)f_2(j_2)$
$[q_1 > 0, y_1 \in \mathcal{Y}_1, q_2 > 0]$	1 Txs, 2 cant Tx	$[q_1 + j_1 - 1, y_1 + 1, q_2 + j_2]$	$f_1(j_1)f_2(j_2)$

$f_i(j_i) = \text{Prob}(j_i \text{ arrivals at node } i \text{ in a slot}) \quad j_i = 0, 1, 2, \cdots$

of the p_i persistent protocol. The lower bounds are obtained by assuming no further arrivals to the system as in the analysis of the p_i persistent protocol. To obtain upper bounds, the auxiliary system will be a time-division multiplexed system in which the frame length is m slots and the number of slots available to node 2 is $m - n$. If $m - n = 1$, the system can be treated as a slotted system with deterministic service times of m slots. If $m - n > 1$, the auxiliary system will be a G−limited system in which the server goes away on a vacation for n slots and serves the $m - n$ customers on arrival or till the system is empty. The analysis of this system is available in [15].

Obtaining the bounds on the average queue lengths is also similar. The transition probabilities for the system in which the packet arrivals per slot are independent are shown in Table V.

D. Numerical Results

The upper and lower bounds on the average system delay and queue lengths at the slot boundaries for two node network using the p_i persistent protocol are evaluated for various loads. Since we have noticed that the discrepancy between the Bernoulli approximation results and the simulation results are greater in the proportional loading case, we will use $\lambda_1 = 2\lambda_2$ in our systems. The values of the p_1 were obtained from (5). These bounds are shown in Figs. 3 and 4, together with comparable data obtained from simulation and from the Bernoulli approximation.

Average waiting times are shown in Fig. 3. In this case, we have adopted the discrete time Bernoulli-per-slot arrival process we used in Section III-A in all cases. Note that we have not included any the slot synchronization time in any of these curves. Fig. 4 shows average queue lengths at slot boundaries, assuming a continuous time Poisson arrival process.

Comparing the simulation results with the bounds in Figs. 3 and 4, we find that the lower bound is extremely tight whereas the upper bound is fairly loose. This is because the auxiliary system that was derived from a worst case situation and hence presents an extremely loose upper bound on the average cumulative delays and queue lengths. Moreover, the moments of the busy period in an M/G/1 queue grow much faster than the moments of the system delay. This yields extremely large values for the coefficients γ_1, γ_2 and δ especially for high loads which makes the upper bound even looser at high loads.

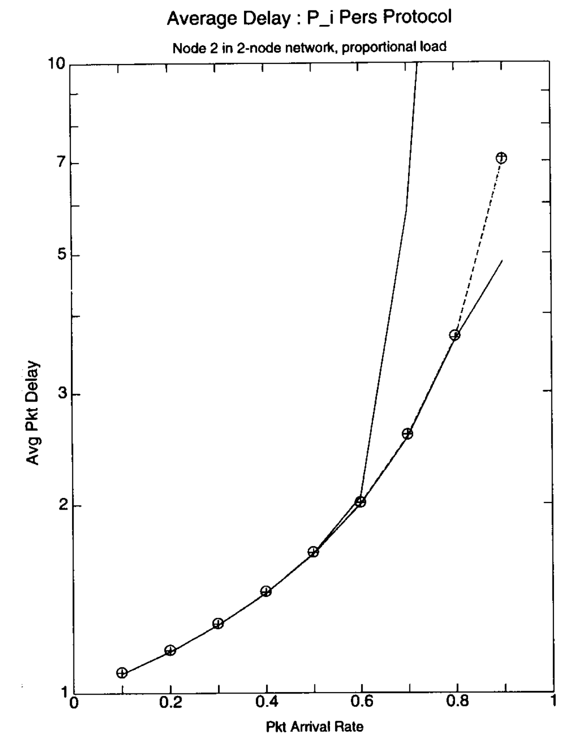


Fig. 3. Upper and lower bounds on the average waiting time in node 2 in a two-node p_i persistent network under a Bernoulli-per-slot traffic model. Solid lines indicate the bounds. Also shown are simulation results (indicated by "+") and the Bernoulli approximation (indicated by "o").

A better selection of these constants will improve the upper bound considerably.

IV. DISTRIBUTED SCHEDULING IS INHERENTLY NON WORK CONSERVING

The original motivation for introducing distributed scheduling algorithms, like the p_i persistent protocol, was to eliminate the bandwidth wasted in cycle-based round robin schemes like Expressnet and Fasnet. These cycle-based schemes are unsuitable for high speed networks, because they require end-to-end feedback that one cycle has been completed before advancing to the next. Thus, it is inevitable that their performance (in

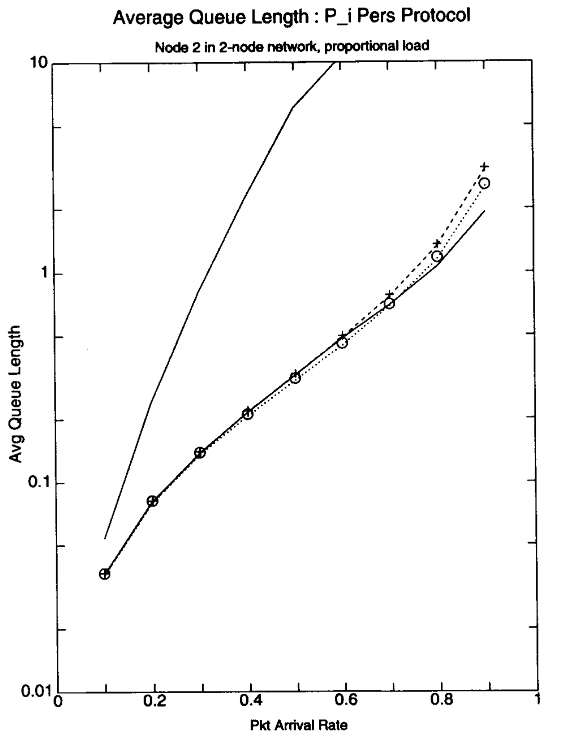


Fig. 4. Upper and lower bounds on the average waiting time in node 2 in a two-node p_i persistent network under a Poisson traffic model. Solid lines indicate the bounds. Also shown are simulation results (indicated by "+") and the Bernoulli approximation (indicated by "o").

TABLE VI
PROBABILITY OF WASTING SLOTS IN THE p_i PERSISTENT AND DETERMINISTIC PROTOCOLS

λ	Prob(slot is wasted)			
	Uniform loading		Proportional Loading	
	p_i pers	n-of-m det	p_i pers	n-of-m det
0.2	0.0187	0.0163	0.0201	0.0176
0.5	0.1107	0.0751	0.1153	0.0797
0.8	0.1578	0.1134	0.1574	0.1152

terms of maximum throughput and/or response times) must degrade as the bandwidth-distance product for the network increases.

Although neither the p_i persistent protocol nor the n-out-of-m protocol suffer from this type of cycle-based overhead, a significant number of slots are still being wasted because this type of distributed scheduling algorithm is not *work conserving*. In other words, the average performance across all nodes is significantly worse than an ideal scheduling policy with access to global information because many of the slots are empty when they reach the tail of the bus, even though there were packets in the system ready for transmission in those slots.

Define wasted bandwidth, P_w, as the proportion of slots that were not used by any node even though there were packets available for transmission. From our simulation model, we obtained the probability that a slot is wasted in both the p_i persistent and the deterministic n-out-of-m protocol for various combinations of loading. These results are shown in Table VI. As can be seen, the p_i persistent protocol has not achieved its goal of preventing wastage of bandwidth (but at least the remaining overhead is independent of the propagation delays). The deterministic n-out-of-m protocol also wastes a some bandwidth although not as much as the p_i persistent protocol.

The explanation for this inefficiency is that the position-dependent flow control mechanisms by which this type of distributed scheduling algorithm achieves fairness are inherently wasteful. In effect, they force the upstream nodes to blindly give up bandwidth that they could have used without knowing whether or not the downstream nodes are able to use it. Consequently, the inevitable large increases in the delay for the upstream nodes do not translate into significant reductions in the delay for the downstream nodes, i.e., they end up punishing the upstream nodes until all the nodes are equally miserable! Indeed, it is easy to demonstrate that no distributed scheduling algorithm for a passive unidirectional bus without upstream feedback can ever equalize the average packet waiting times at all nodes without also increasing the overall average beyond the value for a work conserving scheme. We proceed as follows.

First, let us combine the traffic for nodes $1, 2, \cdots, N-1$ into a single upstream node with arrival rate Λ_{N-1}, node $N-1$ say. Since the combined node can simulate the actions of $N-1$ nodes at no cost in channel overhead (but not conversely), this change can only help reduce delay experienced by the upstream traffic. Also, since there is no feedback in the upstream direction on a unidirectional bus system, the operation of node $N-1$ cannot depend on the state of node N.

To model this type of distributed scheduling algorithm, we will assume that node $N-1$, when its queue is non empty, uses only a fraction p of the arriving empty slots, where p is an optimization parameter. Thus, viewing the upstream node as a queuing system, we see that its service time satisfies $\overline{x_{N-1}} = \frac{1}{p}$ and $\overline{x^2_{N-1}} \geq \overline{x_{N-1}}^2$. (Notice that equality occurs if and only if all service times are deterministic, in which case the upstream node is acting like a Cruz-type regulator for the upstream traffic.) Thus, assuming Poisson traffic, the best policy from node $N-1$'s perspective, for minimizing its own delay for the given value of p, occurs when node $N-1$ acts like a slotted M/D/1 queue using $\overline{x_{N-1}}$ as the slot size. In this case, $\overline{d_{N-1}}$, the average packet delay for node $N-1$, can be obtained from (3) after substituting $\overline{x^2_{N-1}} = \overline{x_{N-1}}^2$. Note that $\overline{x_{N-1}}$ increases monotonically as we impose stricter controls on node $N-1$ by decreasing the parameter p from unity down to Λ_{N-1}, and hence so does $\overline{d_{N-1}}$.

Now consider node N, which we view as a queuing system with vacations. Let us partition the slots arriving from node $N-1$ into a sequence of *service epochs*, each consisting of a free slot followed by the sequence of zero or more consecutive busy slots up to (but not including) the next free slot. Each service epoch can be used to represent a vacation (since if we miss the initial free slot, we must wait until the next epoch), or a service time in our waiting time calculation (since the server will be unable to accept another customer until the end of the epoch). However, in calculating the total time in system for a packet, we must remember to include only the first slot of the epoch in which it departs from the system, because that is where its transmission actually takes place.

Clearly, since node $N-1$ occupies a fraction Λ_{N-1} of the available slots, a simple counting argument can be used to show that the overall average duration of a service epoch must be $\frac{1}{1-\Lambda_{N-1}}$. However, we have not yet specified any other properties of the service epoch sequence, such as the higher moments of the epoch length distribution or its autocorrelation statistics. We will seek a lower bound to $\overline{d_N}$, the average system time at node N, and make the following optimistic assumptions. Since the service epochs are determined by the output process from node $N-1$, let us consider a regenerative cycle for node $N-1$ from the start of one busy period to the start of the next. During each busy period from node $N-1$, which has an average duration $\overline{Y_{N-1}}$, where

$$\overline{Y_{N-1}} = \frac{\overline{x_{N-1}}}{1-\Lambda_{N-1}\overline{x_{N-1}}},$$

node $N-1$ generates free slots at the rate of $1-p$, plus one extra at the very end. Thus, on average, each busy period from node $N-1$ contributes $(1-p)\overline{Y_{N-1}}+1$ free slots per cycle, and to minimize the value of $\overline{x_N^2}$ we will assume they occur deterministically, once every

$$\overline{x_{N,L}} = \frac{\overline{Y_{N-1}}+1}{\overline{Y_{N-1}}(1-p)+1}$$

slots. Similarly, each of the remaining idle slots in the cycle contributes one free slot every $\overline{x_{N,S}} = 1$ slots to the total. Thus, we will assume that

$$\begin{aligned}\overline{x_N^2} &= \frac{\overline{x_{N,L}^2}(\overline{Y_{N-1}}(1-p)+1)+\overline{x_{N,S}^2}(1/\Lambda_{N-1}-1)}{\overline{Y_{N-1}}(1-p)+1/\Lambda_{N-1}}\\ &= \frac{1+\Lambda_{N-1}(p-\Lambda_{N-1})}{(1-\Lambda_{N-1})^2}\end{aligned} \tag{18}$$

Recognizing that $p_N = 1$, it should be clear that our lower bound estimate for $\overline{x_N^2}$ in (18) is always smaller than the corresponding result in (2) for all values of the parameter p, and as $p \to \Lambda_{N-1}$ it converges to $\overline{x_N}^2$ which is the smallest possible value. Thus, substituting (18) into (3), we obtain our lower bound for $\overline{d_N}$ for the given value of p.

In Fig. 5, we have plotted our lower bounds to $\overline{d_{N-1}}$ and $\overline{d_N}$ as a function of the idealized flow control parameter, p, which is applied to the upstream node. The bound on $\overline{d_{N-1}}$ is fairly tight, since it is basically just our deterministic scheme without worrying about the necessity to use only integer-valued service times. The bound on $\overline{d_N}$ is much weaker, since we have ignored the effects due to variability in the node $N-1$ busy period length and the bunching together of the short service epochs. For example, in the limit of $p \to 1$ the system reduces to a two-class HOL priority queuing system, in which case we can show that the correct value of $\overline{d_N}$ is given by $1+\frac{1}{(1-\Lambda_{N-1})(1-\Lambda_N)} = 9.93$, instead of the value 4.0 from our lower bound. Nevertheless, the intersection point for the two system time curves is above the corresponding result for a slotted M/D/1 queue, indicating that some increase in delay is inevitable using this type of policy for achieving fairness.

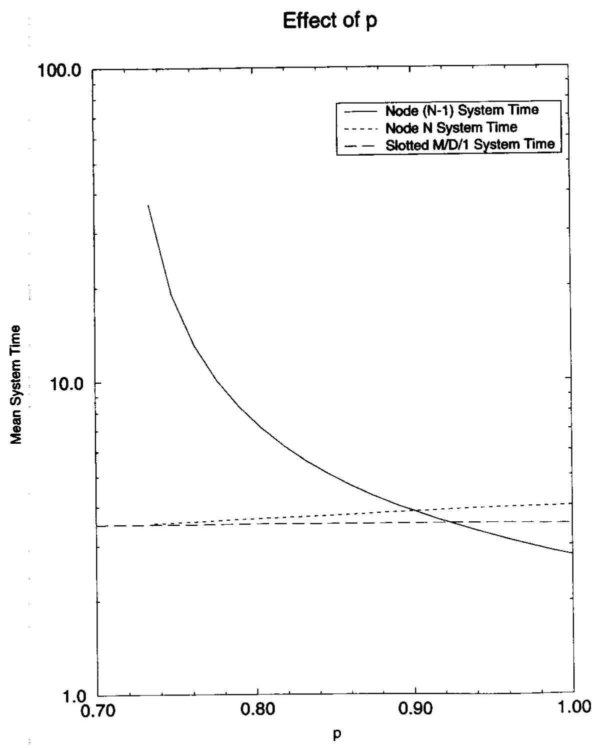


Fig. 5. The effect of the parameter p on our lower bounds to $\overline{d_{N-1}}$ and $\overline{d_N}$, assuming $N = 10$, $\lambda = 0.8$, and uniform loading. Also shown is $\overline{d}$ for a slotted M/D/1 queue, which is also the global average system time for all work conservative policies.

The discussion above has shown that for *every* value of the parameter p, distributed scheduling algorithms, such as the p_i persistent protocol or the n-out-of-m protocol, waste bandwidth. There is, however, an additional practical detail associated with these protocols: the expression for p_i in (18) depends on information about downstream traffic that is not visible to node i. The significance of this detail is that the protocol, which is *intended* to operate on a unidirectional bus *without* feedback, cannot operate without timely access to traffic statistics visible only at the end of the bus, which must be sent in the reverse direction, or delayed for a round-trip time like a cycle-based scheme.

V. The Work Conserving p_i Preemptive Protocol

In this section, we introduce a new type of protocol for unidirectional bus networks that is both *work conserving* and *fair*. That is, the average packet delay across the entire system is the same as it would be for a centralized single server queue. In addition, the average packet delay for each node is (approximately) equal to that global average.

Since we have just proven that no distributed scheduling algorithm, such as the p_i persistent protocol, can be both work conserving and fair, it is obvious that our protocol must be exploiting some side information about the way this type of network is constructed beyond what we considered in the previous section. The extra information is that it is much easier to construct a unidirectional bus network as a sequence of point-to-point channels linking active logic elements at each node (much like a token ring or slotted ring) than as a single passive bus [10]. One reason for this improvement is the high cost and lack of availability of low loss taps for a high speed passive fiber. More importantly, however, since each receiver on a point-to-point system is always listening to a data stream generated by the same transmitter, we avoid the issue of having to re-acquire the data stream each time the identity of the sender changes. Thus, we believe that our decision to assume that the fiber is divided into point-to-point segments between active interfaces is not a technological disadvantage over earlier protocols, as long as the operation of each network interface is simple enough.

The basic idea behind our protocol is to allow each node to act fairly using the traffic information that is visible to it, i.e., its own traffic and the traffic arriving from upstream nodes. Thus, the node assumes that the upstream nodes have already allocated the busy slots among themselves fairly, and then interleaves its own packets among the packets arriving from upstream in such a manner that everyone's average packet delay, upon leaving this node, will again be equalized.

The channel access algorithm followed by node i is as follows. The node maintains two FIFO queues, one for storing traffic arriving from upstream and another for its own packets. At the start of each slot, the node examines the busy/idle status of the incoming slot, and begins copying the arriving packet (if any) into the first free location in the upstream queue. Thereafter, the node selects a packet (if any are available) to be written in the next outgoing slot. If neither queue has any packets (including a newly arriving packet from upstream, since we allow the node to do cut-through switching), the outgoing slot will be empty. If only one of the queues has any packets, the first packet from that queue is sent. Finally, if both of the queues have packets, then the node tosses a coin with bias p_i and transmits the first packet from its own queue if the toss comes up *heads*, and the first packet from the upstream queue otherwise.[1]

Notice that the above access algorithm is work conserving, since a node never generates an empty outgoing slot unless both of its queues are empty. Also, the minimum forwarding delay at a node is on the order of one bit time, since the decision to apply cut-through switching to an incoming packet can be made before the arrival of the address field, using only the status of the busy bit. Furthermore, it should be obvious that we can achieve fairness through a suitable choice of p_i, since the average packet delay at node i is a continuous function of p_i, and the boundary cases of $p_i = \{0, 1\}$ reduce the system to a strict priority queue with node i as the lowest or highest class, respectively. Thus, it remains to describe an algorithm for calculating p_i as a function of λ_i and Λ_{i-1}. This will show that our algorithm is implementable using only information easily obtainable by monitoring the incoming channel.

[1] A random coin toss was chosen (instead of the deterministic n-out-of-m rule) for simplicity, since our goal is to demonstrate the fundamental advantages offered by an active tap. This choice may also make it easier to allow dynamic adjustment of p_i.

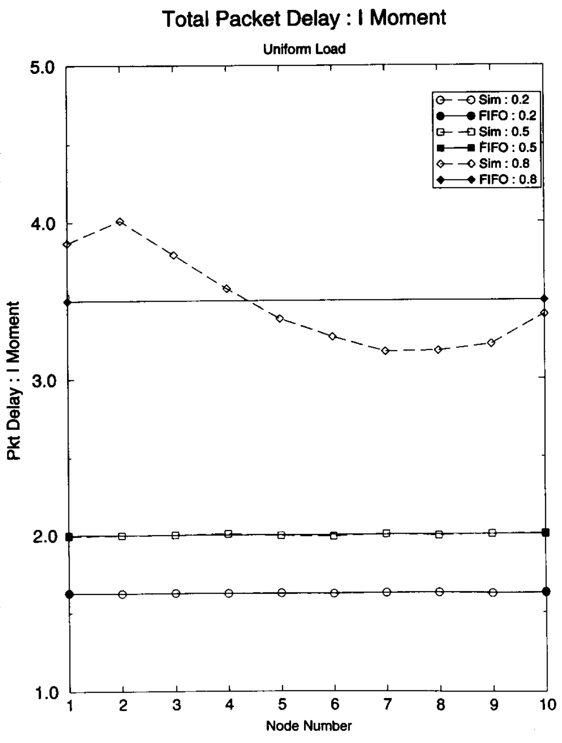


Fig. 6. Total packet delay at each node in the p_i preemptive protocol for a 10-node network with uniform loading.

Our goal in tuning the access protocol is to adjust p_i until the average packet delay experienced by packets originating at node i up to their departure from node i is equal to $\overline{D_i}$, the average packet delay, in a single node system with packet arrival rate Λ_i. Fortunately, this average is very easy to calculate using standard results for an M/D/1 queue with two HOL priority classes. We simply aggregate the traffic from nodes $1, \cdots, i$ to form the high priority class, and treat empty slots as the (saturated) low priority class. Hence,

$$\overline{D_i} = \frac{1}{2(1 - \Lambda_i)}. \tag{19}$$

Because our system is work conserving (and all packets are of fixed length), we know that the global average delay for traffic originating at all nodes must be invariant across all scheduling policies. Thus, since the average delay for node i packets is $\overline{D_i}$ by construction, the overall average packet delay accumulated by all packets from nodes $1, \cdots, i-1$ up to their departure from node i must also be $\overline{D_i}$. But since the scheduling decisions at each of the upstream nodes are done fairly, using this same algorithm, we can conclude that for *any* node $j \leq i$, the average packet delay for node j traffic up to its departure from node i must be equal to $\overline{D_i}$.

Unfortunately, finding the correct value for p_i is not as easy as it looks, and we tried many seemingly reasonable approaches before finding one that gave acceptable results. Our approach to estimating p_i will be to simplify the derivation by

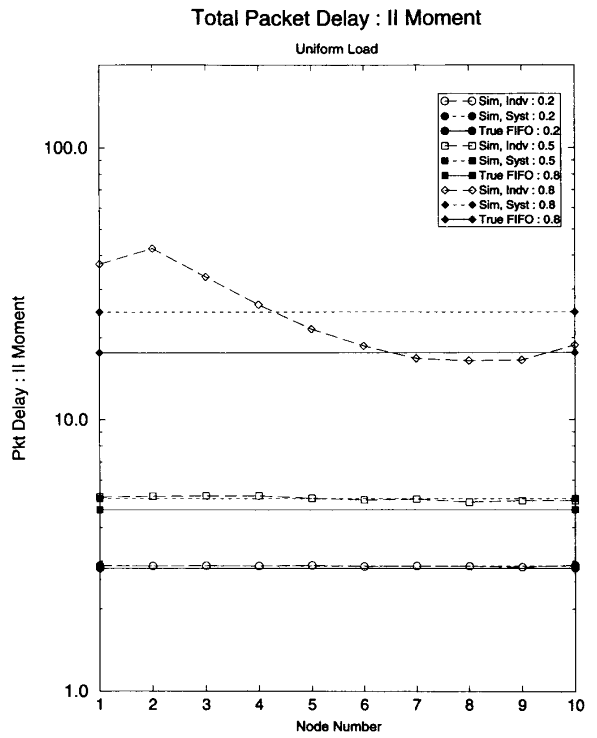


Fig. 7. Second moment of the total packet delay at each node in the p_i preemptive protocol for a 10-node network with uniform loading.

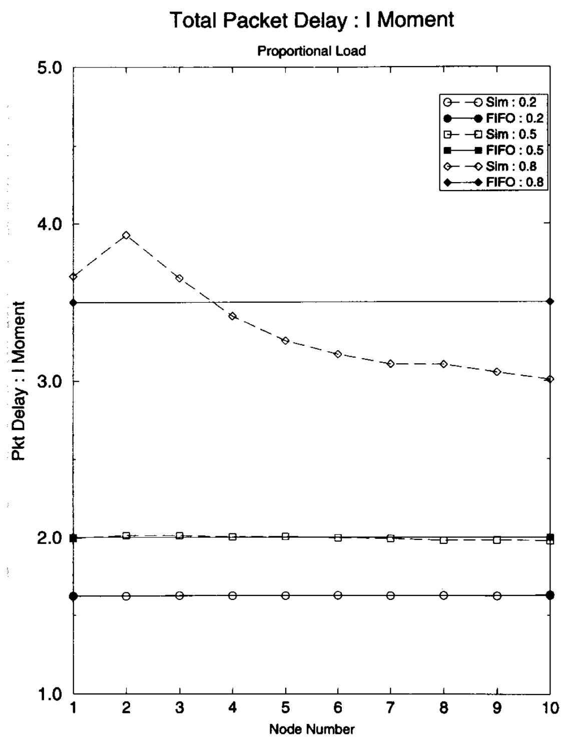


Fig. 8. Total packet delay at each node in the p_i preemptive protocol for a 10-node network with proportional loading.

partitioning the problem into two stages. First, we identify the *target arrival rate*, γ_{i-1}, for a source of high priority traffic, whose insertion at node i would increase the average packet delay for the lower priority node i traffic up to $\overline{D_i}$. Then, we find p_i using a virtual arrival rate argument, as the constant of proportionality for reducing the actual upstream arrival rate, Λ_{i-1}, down to γ_{i-1}.

For any given value of γ_{i-1}, we can find the average packet delay for node i traffic as the average waiting time, $\overline{D_{i,\gamma}}$, for the middle priority class in an M/D/1 queue with three HOL priority classes, where γ_{i-1} is the high priority arrival rate, λ_i is the middle priority arrival rate, and idle slots form a saturated low priority class. Thus,

$$\overline{D_{i,\gamma}} = \frac{1}{2(1-\gamma_{i-1})(1-\gamma_{i-1}-\lambda_i)}. \tag{20}$$

Equating (19) and (20) and solving for γ_{i-1}, we obtain

$$\gamma_{i-1} = 1 - \frac{\lambda_i}{2} - \sqrt{1-\Lambda_{i-1}-\lambda_i+(\lambda_i/2)^2}. \tag{21}$$

Now, returning to (20), we recognize that the factor $\frac{1}{1-\gamma_{i-1}}$ in the HOL waiting time formula represents the average length of a *delay cycle* where the server is busy clearing the queue of high priority customers (who are arriving at rate γ_{i-1}) after serving a single normal customer. Obviously, if we simply let the upstream packets be our high priority class, then the high priority arrival rate would be Λ_{i-1}, which is too large for our needs, but we could achieve the desired behavior by discarding a fraction p_i of the upstream packets, where

$$\gamma_{i-1} = (1-p_i)\Lambda_1$$

or

$$p_i = 1 - \frac{\gamma_{i-1}}{\Lambda_{i-1}}. \tag{22}$$

Of course, node i is not allowed to throw away any of the upstream traffic for its own convenience, just preempt it! Thus, we will use p_i from (22) as the preemption probability in our protocol. For small values of Λ_{i-1}, this heuristic should be quite accurate, since the upstream queue length is unlikely to be very large, and there is little difference between independently postponing *each* or collectively postponing *all* of the available high priority packets until another node i packet has been sent. However, as Λ_{i-1} increases, this argument breaks down, and we expect this heuristic to allow node i to preempt the upstream traffic too often.

Using the value of p_i obtained from (22), we simulated a 10-node network with Poisson packet arrivals. Our results are presented in Figs. 6 and 7 for loads uniformly distributed along the bus, and in Figs. 8 and 9 for proportionally distributed along the bus. For $\Lambda_{10} = 0.2$, 0.5 and 0.8, we plot the first and second moments of the total delay, excluding propagation delay, for a packet to reach the end of the bus. The corresponding values for true work conserving FIFO service are also shown for comparison. As is evident from the graphs, our protocol does a good job of achieving its goal of fairness without added delays. Not only does the global mean for the total delay in our

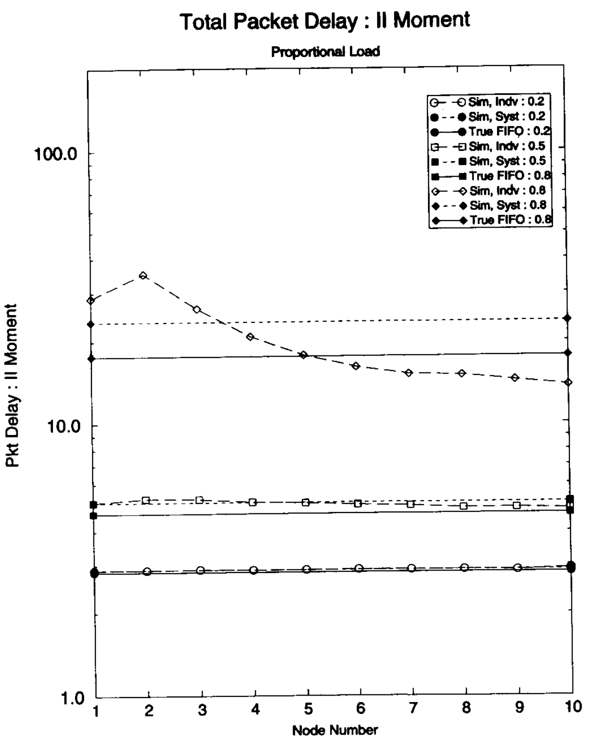


Fig. 9. Second moment of the total packet delay at each node in the p_i preemptive protocol for a 10-node network with proportional loading.

protocol match the value for a true FIFO queue, the variability among the nodes is negligible for $\Lambda_{10} = 0.2$ and 0.5, and is usually within 10 percent of the global mean for $\Lambda_{10} = 0.8$.

VI. Discussion and Conclusion

In this paper, we have described several contributions to the design and analysis of access protocols for shared medium unidirectional bus networks. First, we have found an exact method for calculating average packet delays and queue lengths in the well-known p_i persistent protocol. Then, we found a way to improve the delay characteristics of the p_i persistent protocol itself, by replacing random coin tosses by a periodic algorithm, which is a generalization of the counting algorithm from the Bandwidth Balancing Algorithm in DQDB. For any given choice of $\{p_i\}$, and hence *average* access delay, this modification reduces the corresponding *variance* of the access delay. Thus, our modified access scheme, which we call the deterministic n-out-of-m protocol, achieves a reduction in both the mean and second moment of the delay at all nodes in comparison to the p_i persistent protocol.

In spite of the delay improvements under the deterministic n-out-of-m protocol, the global average packet delay across all nodes is still higher than it would be in a centralized single queue environment. We also provide some direct simulation measurements to document the fact that both the original p_i persistent protocol and our improved deterministic n-out-of-m protocol still waste some of the available bandwidth to achieve fairness. Moreover, we present a lower bounding argument to show that no distributed scheduling algorithm operating without feedback in the upstream direction can match the average response times of a work conserving queue.

In the last part of the paper, we introduce a new class of access protocols for unidirectional bus networks that is both work conserving and fair. This class circumvents the above lower bounding argument by assuming that the bus is implemented as a series of point-to-point segments connecting active interfaces at each node, and we argue that this assumption may actually be less expensive to implement than a passive bus with multiple taps. Because of their active interfaces, the nodes in our network need not be repeating the same packet on the outgoing segment that is arriving on the incoming segment. Thus, downstream nodes can insert their own packets among those already sent by the upstream nodes in order to equalize their respective delays. As an example, we describe the p_i preemptive protocol in considerable detail. We show that the required functionality of its active network interfaces is extremely simple, and that only a single bit-time delay per node is sufficient. We also show that at any node, both normal operation of the protocol and periodic updates to its preemption probability, p_i, only depend on the upstream traffic. The only difficulty in the analysis of the p_i preemptive protocol is in determining the appropriate preemption probability for each node. We present a simple and effective heuristic for calculating p_i, and use simulation to show that it does a good job of equalizing the first two moments of the packet delay for each node.

Perhaps the most significant contribution in this paper is to highlight the importance of the seemingly minor implementation detail about whether or not to use active network interfaces at each node. Thus, we believe that the p_i preemptive protocol represents an important advance in the area of access schemes for unidirectional bus networks.

References

[1] B. T. Doshi, "A note on stochastic decomposition in a GI/G/1 queue with vacations or set-up times," *J. Appl. Probability*, vol. 22, pp. 419–428, Jun. 1985.
[2] L. Georgiades and P. Papantoni-Kazakos, "Limited sensing algorithm for the packet broadcast channel," *IEEE Trans. Info. Theory*, vol. IT-31, pp. 280–294, Mar. 1985.
[3] L. Georgiades, L. F. Merakos, and P. Papantoni-Kazakos, "A method for the delay analysis of random multiple access algorithms whose delay process is regenerative," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp1051-1062, July 1987.
[4] E. L. Hahne, A. K. Choudhury, and N. F. Maxemchuk, "DQDB Networks with and without bandwidth balancing," *IEEE Trans. Commun.*, vol. 40, pp. 1192–1204, July 1992.
[5] M. Hofri and Z. Rosberg, "Packet delay under the golden ratio weighted TDM policy in a multiple access channel," *IEEE Trans. Info. Theory*, vol. IT-33, pp. 341–349, May 1987.
[6] A. Itai and Z. Rosberg, "A Golden Ratio Control Policy for a Multiple-Access Channel," *IEEE Trans. Automat. Control*, vol. AC-29, pp. 712–718, Aug. 1984.
[7] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*. New York: Interscience, 1958.
[8] L. Kleinrock, *Queuing Systems, Vol. II: Computer Applications*. New York: Wiley, 1975
[9] H. Levy and L. Kleinrock, "A queue with starter and a queue with vacations: Delay analysis by decomposition," *Oper. Res.*, vol. 34, no. 3, pp. 426–436, May-June 1986.
[10] Private conversations with N. F. Maxemchuk.
[11] G. J. Miller, "Design and performance evaluation of a dynamic priority mechanism for high speed unidirectional bus networks," Ph. D.

dissertation, CIS Department, University of Delaware, 1993.

[12] B. Mukherjee and J. S. Meditch, "The p_i–persistent protocol for unidirectional broadcast bus networks," *IEEE Trans. Commun.*, vol. COM-36, pp. 1277–1286, Dec. 1988.

[13] B. Mukherjee, "Algorithm for the p_i-persistent protocol for high speed fiber optic networks," *Comp. Commun.*, vol. 13, no. 7, pp. 387–398, Sept. 1990.

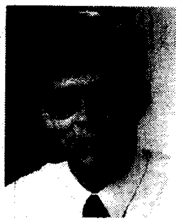
[14] W. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics.* New York: McGraw Hill, 1979.

[15] H. Takagi, *Queuing Analysis*, vol. 1. Amsterdam, The Netherlands: North Holland, 1991.

[16] N. D. Vvedenskaya and B. S. Tsybakhov, "Packet delay in the case of a multiple access stack algorithm," *Problems Info. Transmission*, vol. 20, no. 2, pp. 137–153, 1984.

D. Manjunath (S'85–M'92) received the B. E. degree from Mysore University, the M. S. degree from the Indian Institute of Technology, Madras, and the Ph. D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1986, 1989, and 1993, respectively.

He worked in the Corporate R&D Center of General Electric in Schenectady, NY during the summer of 1990. He was a visiting faculty in the Computer and Information Sciences Department of the University of Delaware and a Postdoctoral Fellow in the Computer Science Department of the University of Toronto. He has been on the Electrical Engineering faculty of the Indian Institute of Technology, Kampur, since December 1994. His research interests are in communication networks, performance analysis of systems, queuing systems, and multimedia communications.



Mart L. Molle (S'80–M'82) was born in Toronto, Canada on November 2, 1953. He received the B. Sc. (Hons.) degree in mathematics/computer science from Queen's University at Kingston, Canada, in 1976, and the M. S. and Ph. D. degrees in computer science from the University of California, Los Angeles, in 1978 and 1981, respectively.

In 1981 he joined the University of Toronto, where, since 1991, he was a Professor of Computer Science, with a cross-appointment in Electrical and Computer Engineering. In 1994, he joined the University of California, Riverside, where he is currently a Professor of Computer Science. In 1987–1988, he was a Visiting Associate Professor at the University of California, Irvine. His research interests include the performance evaluation of protocols for computer networks and of distributed systems. He is particularly interested in analytical modeling techniques (especially those that can be solved efficiently), fundamental performance limits, applications of queuing theory and scheduling theory to distributed algorithms, and model validation through measurement and simulation. He has published extensively on the performance evaluation of various local area network access protocols and conflict resolution algorithms.

Dr. Molle is an editor for the IEEE/ACM Transactions on Networking, and chairs a task force in the IEEE 802.3 Ethernet Standards Committee. He is also a member of the ACM.