Spatial TDMA: A Collision-Free Multihop Channel Access Protocol

RANDOLPH NELSON, MEMBER, IEEE, AND LEONARD KLEINROCK, FELLOW, IEEE

Abstract—**In this paper we define a broadcast channel access protocol called *spatial TDMA*, which is designed specifically to operate in a mutlihop packet radio environment where the location of the nodes of the network is assumed to be fixed. The defined protocol assigns transmission rights to nodes in the network in a local TDMA fashion and is collision-free. Methods for determining slot allocations are developed, and an approximate solution is given for determining the assignment of capacities for the links of the network that minimizes the average delay of messages in the system.**

I. INTRODUCTION

NETWORKS have traditionally consisted of a set of switches connected together by some form of cable. Because cable must be strung point-to-point and requires land acquisition and construction of supporting structures, the design of such networks requires careful, long-range planning, and it is not surprising that considerable research has been devoted to the problems of this initial design. An example of a problem that has been extensively studied arises when the locations of the nodes of the network and their traffic characteristics are assumed to be known. In this case, one must determine a procedure for choosing the capacities of the communication lines. Since the cost of the network is an increasing function of these capacities, it is important that the designer of such a network selects the assignment of the capacity of the links of the network to minimize network expense, while preserving a tolerable delay for messages in the system. Various solutions [1]–[4] have been proposed for this, and other problems of a similar genre, which are commonly classified as *capacity assignment problems*. As one can expect, however, network specifications often change, and an optimal design for an initial network may be far from optimal after changes are made either to the traffic characteristics of the nodes or to the network's topological structure. Indeed, even the minor change of optimally adding one new node to a wire network can be a formidable problem. Besides the difficulty of connecting cable from the new node to its adjacent neighbors, one must also solve the capacity assignment problem again for the changed network. If the traffic offered by the new node changes the loads on the links of the already existing network, then some of the links will need to be upgraded to higher capacities. Likewise, other links might be able to have their capacities reduced if some of the flow through them can be routed over the new lines. Upgrading existing lines, however, is expensive, and the cost of preserving this optimality in the network after the addition of a new node could be so formidable that the designer would have to settle for a less than optimal solution. As the process of adding new nodes to the network continued, it would not be surprising to see performance seriously deteriorate.

To avoid these problems, one needs to use a more flexible medium for interconnecting the switches of the network. A broadcast medium such as radio offers flexibility to topological changes, and has the property that capacities can be changed to reflect alterations in the specifications of the network. The problem of readjusting channel capacities in an optimal manner is not straightforward, and depends upon the channel access protocol used by the nodes of the network. In this paper we propose a collision-free channel access protocol for a packet radio. The protocol operates in a packet radio environment in which the location of the nodes of the network are assumed to be fixed and known. We also assume that time is divided into slots equal in length to the amount of time it takes to transmit one packet over the channel, and that all nodes are synchronized as to slot boundaries. We also assume that nodes are synchronized as to the beginning of time cycles, which are defined later in the paper. Because the protocol is collision-free, calculating the capacity of the channels linking the nodes of the network is straightforward, and this capacity can be changed in accordance with changing network conditions. An outline of the paper is as follows. In Section II of the paper we describe the protocol, and in Section III we formulate a fluid approximation to determine its delay characteristics and compare the approximation to simulation results. In Section IV we formulate a mathematical program that is used to solve the capacity assignment problem for networks using this protocol.

II. DESCRIPTION OF THE PROTOCOL

Spatial TDMA is a generalization of the TDMA protocol for multihop networks. Each frame of the protocol consists of a number of slots that are allocated to a set of noninterfering transmissions in the network. To best see the operation of the protocol, consider the network shown in Fig. 1. In this figure, a directed arc from node i to node j indicates that node j can hear a transmission from node i. When a transmission takes place over an arc of the network, we say that arc is *enabled*. This happens when a node transmits a message addressed towards its neighboring node that is incident on the directed arc. From this graph one can see that if node 2 transmitted to node 1 (arc 2 is enabled), 1's reception would not be interfered with if node 5 also was sending a message to node 6. One could thus conclude that simultaneous enabling of arcs 2 and 9, as labeled in the figure, does not result in a collision at nodes 1 or 6. In order to formalize this observation, we define a compatibility matrix which is a symmetric binary $(m \times m)$ where m is the number of arcs in the directed graph. A 1 in the (i, j) position of the compatibility matrix indicates that arcs i and j can be simultaneously enabled without causing a collision at either of their respective destinations in the network. The compati-

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication without oral presentation. Manuscript received October 4, 1982; revised October 2, 1984.

R. Nelson is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

L. Kleinrock is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

0090-6778/85/0900-0934$01.00

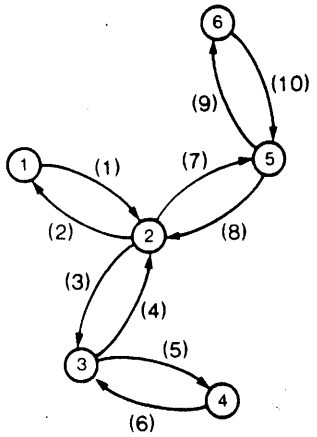


Fig. 1. A sample network.

bility matrix for the network of Fig. 1 is given by

$$CM = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Using this matrix, one can generate a set of *cliques,* containing arcs having the property that all arcs in the same clique can be simultaneously enabled without causing any collision in the network. This is done by using the matrix *CM* as if it were the adjacency matrix for a graph and then running maximal cliques algorithms on this graph. If we let C_i denote the ith clique, we can form a *clique cover, C,* which is a set of maximal cliques $C = \{C_1, C_2, \cdots, C_k\}$ having the property that every arc of the network is contained in at least one member of C. *Note that one possible clique cover is the set of all maximal cliques.*

Spatial TDMA works in the following manner. For each clique C_i in a given clique cover C, we assign an integral number of slots t_i from a given fixed total number of T slots that make up a cycle that repeats over time. This cycle is very much like a TDMA frame in that if the vth slot of the cycle is assigned to clique i, i.e., is one of the t_i slots, then all arcs in clique i are allowed to be enabled during the vth slot. From the way cliques are assigned slots in a cycle, collisions are avoided at all the recipients of these packets. Each slot from the cycle is assigned to a unique clique from the clique cover. Since any particular arc can be contained in more than one clique, the times during which an arc is enabled depend upon the slots assigned to the cliques of which it is a member. Thus, the capacity of an arc in a spatial TDMA network depends upon the total amount of time from the cycle that the arc is assigned permission to transmit. For example, one clique cover for the network of Fig. 1 is given by

$$C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$$

where

$$C_1 = \{1, 6, 10\} \quad C_2 = \{2, 5, 9\} \quad C_3 = \{3, 9\}$$

$$C_4 = \{4, 10\} \quad C_5 = \{5, 7\} \quad C_6 = \{6, 8\}.$$

Suppose for this clique cover we assign a total of t_i, $i = 1, 2, \cdots, 6$ slots, from a cycle of T slots, for the ith clique of the clique cover. For this assignment, out of T slots of the cycle, arc 5 is enabled for a total of $t_2 + t_5$ slots. We will distinguish the traffic arriving at a given node into classes where there is a class for each possible neighboring node to which the given node can transmit. Thus, node 2 has 3 classes of traffic (i.e., a class for each possible destination node 1, 3, 5). The overall network thus acts like a multiclass open network of queues. In the next section we will create an approximation to the mean system delay for messages in such a network.

It should be mentioned that the t_i slots allocated to the ith clique do not necessarily need to be contiguous within the cycle. In fact, as we will later see, there are good reasons to attempt to distribute slots allocated to one clique uniformly over the duration of the cycle. We should also note that synchronization in a packet radio network implies that each slot is expanded by a guard time which is used to offset the varying geographical distances between nodes. The capacity of the channel lost to this guard band is not taken into account in our calculations of delay performance. Thus, our calculations of the mean system delay are lower than what we would obtain if this lost capacity were accounted for in the analysis.

We should also point out that an alternative to spatial TDMA could be spatial FDMA, in which nonoverlapping frequency bands would be assigned to each clique in the network. This protocol, however, would suffer from lost capacity due to the guard bands separating each frequency band, and would not be as adaptable as spatial TDMA to the addition or deletion of nodes in the network. Additionally, changing the capacity of arcs in the network to adapt to changing traffic patterns would be more difficult using spatial FDMA.

III. Delay Analysis

In this section we develop an approximation to the mean system delay for messages in the network. The network is a multiclass open queueing network where each node has a class of traffic for each of its neighboring nodes to which it can transmit. If we let α_i^c be the amount of traffic per cycle passing through node i of message class c, then $\alpha_i = \Sigma_c \alpha_i^c$ is the total traffic passing through the node i per cycle. This traffic consists of packets being routed through the ith node (the network is a multihop network) and also of arrivals from the attached host computer. Let γ be the average amount of external traffic into the network per cycle and denote $D_i^c(t)$, $t = (t_1, t_2, \cdots, t_k)$ where k is the number of cliques in the clique cover and t_i is the number of slots allocated to clique i from the cycle of length $T = \Sigma_1^k t_i$, to be the average system delay experienced by a class c job passing through node i. The mean system delay of messages in the network is given by

$$D(t) = \sum_i \frac{\alpha_i}{\gamma} \left\{ \sum_c \frac{\alpha_i^c}{\alpha_i} D_i^c(t) \right\}. \tag{1}$$

An exact analysis for the average system delay has not been carried out for (1), and appears to be very difficult. Thus, Section III-A develops an approximation for $D_i^c(t)$ and then Section III-B compares the approximation to simulation results.

A. Queueing Approximation

In this section we will describe an algorithm for approximating the average system delay of a single class of messages passing through a single node using spatial TDMA. To avoid cumbersome notation, we will eliminate the subscript of i and superscript of c on the variables defined in this section, and when we speak of arrivals or departures we implicitly mean those of the given class of messages at the given node under consideration. We note that for t, fixed arrivals of a given

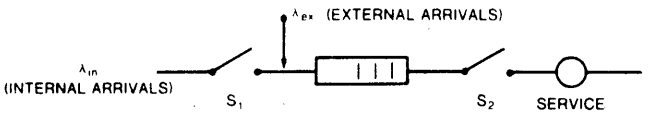


Fig. 2. Model of queue for nodes in the network.

class are independent of those of other classes and do not influence the queueing times of other classes. The queueing system at a given node created by this protocol consists of nonoverlapping internal arrival (arrivals from other nodes of the network), service (times when the node can transmit packets of *that class*), and idle periods (times during which the node cannot receive or transmit packets) which are enabled during specific periods of the cycle. We will call the activity pattern for a particular class at a particular node a *node-class-frame* or simply a *frame* for short. As an example in Fig. 2, we have shown the queueing system that is formed by a single class of packets at a single node of a spatial TDMA network, and also show its corresponding frame (for that class of packets) in Fig. 3. The cycle time for the system shown in Fig. 3 consists of $T = 100$ slots. The queue contains two switches $S1$ and $S2$, which are used to control the internal input and service processes. At most one of these switches can be closed during a given slot of the frame, since we do not allow simultaneous reception and transmission by radios in the network. During the first 20 slots of the frame in Fig. 3, we see an *internal arrival interval*, during which $S1$ is closed ($S2$ will be open), i.e., arrivals enter from inside the network. Note that arrivals from the attached host computer, called *external arrivals*, can also occur during this time. We will model the internal arrival process as Bernoulli process with a probability λ_{in} of having a packet arrive in the slot. This is an approximation to the actual arrival process, which is composed of departures from other nodes in the network. With our assumption for an internal arrival interval of m slots, if we let $P[k \text{ internal}|m]$ be the probability that k internal arrivals occur in m slots, we have

$$P[k \text{ internal}|m] = \binom{m}{k} \lambda_{in}{}^{k}(1-\lambda_{in})^{m-k}$$
$$k = 0, 1, \cdots, m. \tag{2}$$

During the next phase of the frame, a *service interval* of 10 units in which $S2$ is closed ($S1$ is open), packets are served at the rate of one per slot. During this time at most 10 packets can be transmitted. The next phase we show is an *idle interval* during which both $S1$ and $S2$ are open and no internal arrivals or services are allowed. During this time, and over the entire frame, *external messages* (from the node's attached host) can arrive. These are also assumed to arrive from a Bernoulli process and, as shown in Fig. 2, immediately enter the tail of the queue with a probability λ_{ex} of having a packet arrive in a slot. The probability that k external arrivals occur in m slots is given by (2) with λ_{in} replaced by λ_{ex}. In summary, the queueing operation of the network consists of switches $S1$ and $S2$ turning on and off according to the time patterns depicted in Fig. 3 and continuing to cycle every 100 slots.

To approximate the mean system delay we use a fluid approximation [5]. This approximation was found to give good results, and we will illustrate this method using Fig. 4. In the fluid approximation, waiting times are calculated by assuming that the actual, stochastically varying backlog of packets in the queue is approximated by the expected backlog. In Fig. 4 we have plotted the growth of the expected backlog of packets in the system during the course of the frame shown in Fig. 3. For reasons to be explained later, we have started this growth pattern at the beginning of the last idle interval (i.e., at $t = 90$). During this interval, since internal arrivals are prohibited, only external arrivals can add to the backlog. The rate (per slot) at which the queue increases during an idle period is λ_{ex} packets per slot, and in Fig. 4 we see that the expected backlog grows linearly with this slope during this interval. As the frame progresses to the first internal arrival interval, the growth rate for the backlog is given by $\lambda_{ex} + \lambda_{in}$ packets per slot. A service period follows in the next interval and the backlog drops by a rate of $\lambda_{ex} - 1 \leq 0$ packets per slot. As seen in Fig. 4 (points 25–30), the backlog of packets in the queue drops to zero before the service interval is finished. External arrivals during this time are assumed to be immediately serviced and, thus, do not contribute to the backlog. This process continues in this manner until the last idle period, at which point it starts from a zero backlog once again. We will call a point on a node-class-frame a *zero-point* if, starting with an empty queue at this point, it yields an average backlog of zero after exactly one cycle. From the figure it is clear that points in (25–30) and (75–90) are zero points. For a fluid approximation, this implies that all packets that arrived in the frame are serviced (i.e., a zero-point is a regenerative point for the frame). Let $T = \Sigma_{i=1}^{t} t_i$ be the length of the frame, and let T_{id}, T_{in}, and T_s be the total amount of time from the frame for idle, internal arrival, and service periods, respectively. The area under the backlog curve represents the number of packet–seconds accumulated during the frame. Dividing this by the average number of packets that arrive during the frame, $\lambda_{ex}T + \lambda_{in}T_{in}$, gives, by Little's result [6], the average time spent in the queueing system. Because we began the calculation at a zero-point, packet delays for all arrivals to the frame are counted. The utilzation ρ of the system due to that class of customers is equal to the average number of packets entering the system during a cycle divided by the maximum number that could be serviced, and thus, we have $\rho = (\lambda_{ex}T + \lambda_{in}T_{in})/T_s$. It is clear that because of the fluid approximation, we can always find a zero-point on any frame satisfying $\rho < 1$.

This then describes the algorithm for the system delay approximation to (1) for a single class of traffic at a node, which we summarize as follows.

1) Find a zero-point.
2) Calculate the area under the backlog curve.
3) Divide this area by $\lambda_{ex}T + \lambda_{in}T_{in}$ to arrive at the average system delay.

The ordering of the intervals and their lengths, i.e., the choice of t, greatly influence the average system delay. Suppose, for example, that we change the frame in Fig. 3 by coalescing all the service and intervals together and placing them on the frame, as shown in Fig. 5. Although this frame has exactly the same interval lengths (and thus the same utilization), this translation increases the average system delay (in fact, it is a worst case example). In this system, any messages that arrive during the internal arrival interval must wait at least 30 time units before being serviced. If the service and internal arrival intervals are interchanged, it is clear that the average system delay would decrease by at least this much. In fact, the system that has the minimum average system delay is one that spreads arrival and service intervals in infinitesimal units that alternate with each other. In this way, an arrival is immediately serviced in the following interval after accruing little waiting time. In a practical implementation of spatial TDMA there are limits to how small one can make interval sizes, since radios have a finite switching time between transmission and reception. Note that choosing a frame pattern that minimizes the system delay for one particular class at one particular queue in the network does not, in general, decrease the mean system delay for all of the messages in the network. In fact, finding the slot allocation that does achieve the minimum system delay for all messages in the network is a very difficult problem.

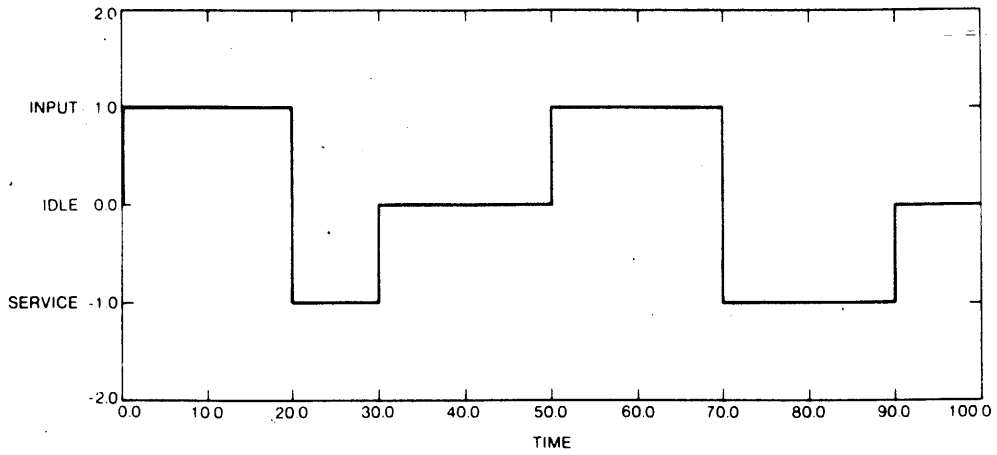


Fig. 3. An example time frame.

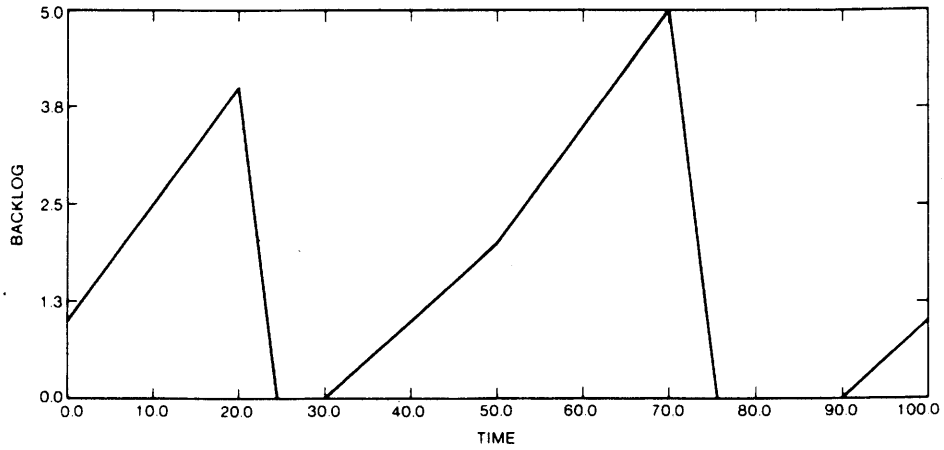


Fig. 4. The backlog for the time frame of Fig. 3.

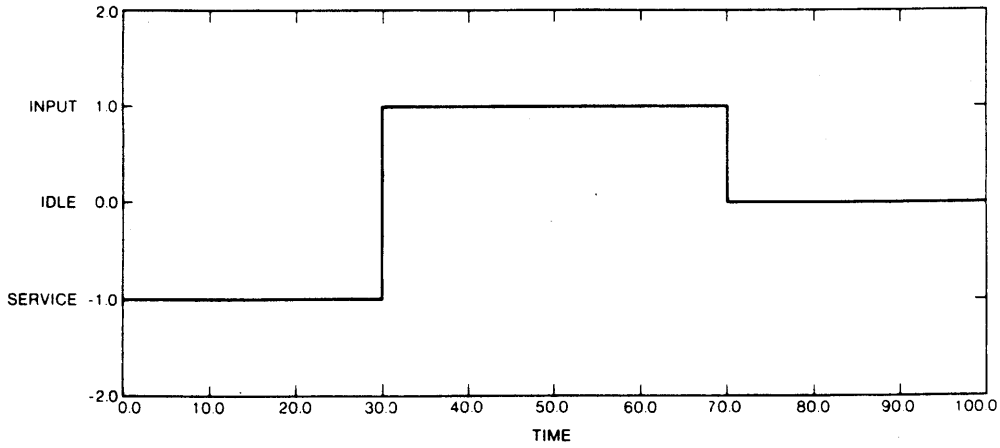
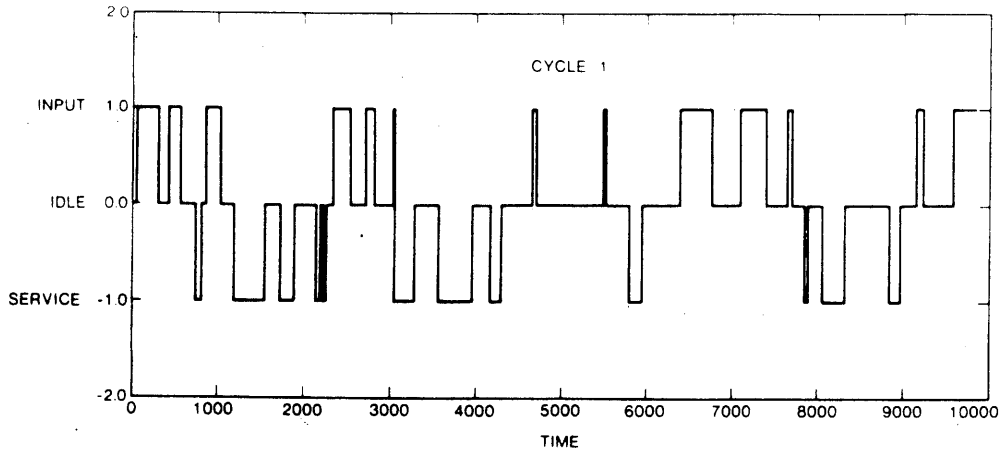


Fig. 5. A worst case time frame.

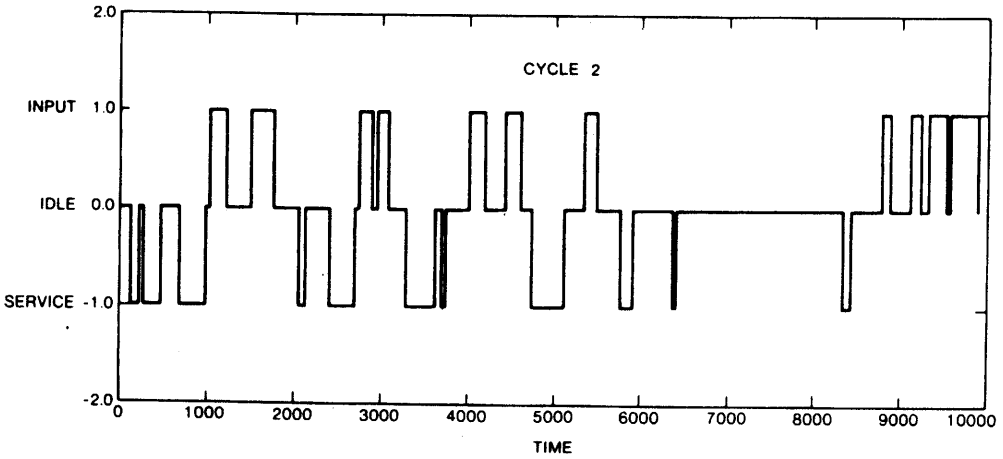
B. Discussion of Delay Results

In this section we compare simulation results to those found using the fluid approximation described in the previous section. The model we simulated consisted of a single spatial TDMA node and we obtained the mean delay of messages of a single class. To do this, we randomly generated many nodal frames that had the same input parameters T, T_{id}, T_{in}, T_s, and λ_{in}, λ_{ex} and results of the simulation checked against those of the approximation. Three such frames are shown in Fig. 6, where $+1$ steps correspond to internal arrival intervals, 0 to idle intervals, and -1 to service intervals. In all three frames $T = 10\,000$, $T_{id} = 6000$, $T_{in} = 2000$, $T_s = 2000$, the average service and internal arrival intervals have length 200, and the average external arrival rate is 0.02 packets per slot. The mean service time as a function of ρ (the utilization was changed by varying the value of λ_{in}) for these frames is shown in Fig. 7.

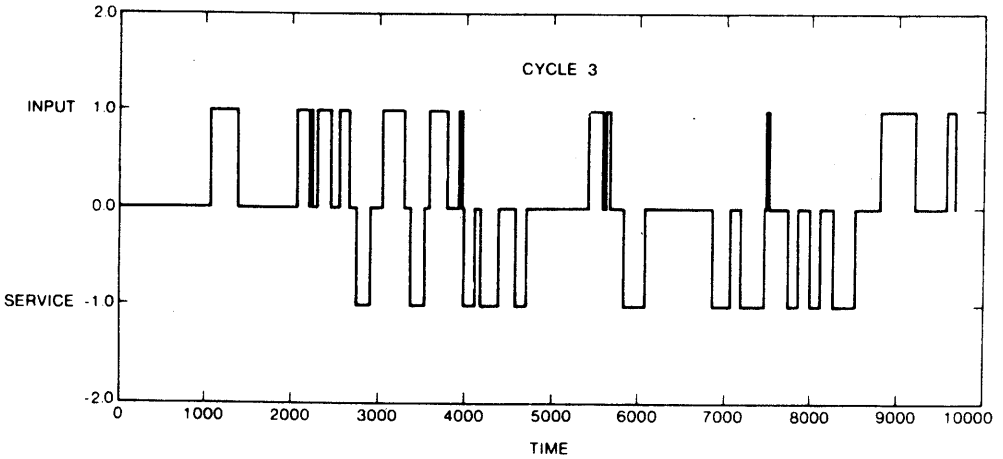
There are several interesting features of these curves. We first see the close match between the simulation points and approximation, given by the solid line, thus assuring us that the fluid approximation is accurate. This close fit for all utilizations was found for all the examples we studied. This accuracy arises from the fact that the variance of the internal arrival (and external) process is limited, since there can be at most one packet arrival per slot. The variation between the



(a)



(b)



(c)

Fig. 6. Three randomly generated time frames.

mean system time for the three frames is very large, which shows the dependency upon the ordering and size of the intervals of the frame. For example, for $\rho = 0.7$, frame 1 has a mean system delay of about 650 slots, whereas frame 3 has a value of 2800, more than four times as much. The extreme delays of the third frame arise from the long periods (9000, 10 000) and (0, 2000), during which there are no service intervals. All packets arriving during these periods create a backlog that cannot be depleted until much later in the frame. On the other hand, the fortuitous placement of intervals in frame 1 consists of groups of arrival intervals followed by service intervals that allow an accumulated backlog to be serviced quickly.

Even though there is a large variance in the curves, there is a similarity in the shapes of the curves. The curves are very well approximated by a piecewise linear function (this curve, which was drawn by hand, is shown as a dashed line in Fig. 7). The slope changes in the piecewise linear approximation occur when the arrival rate is so large that the arrivals to an internal arrival interval cannot be serviced in the next set of service epochs. For example, the change about the point $\rho = 0.6$ for frame 2 arises from the fact that for $\rho > 0.6$, some

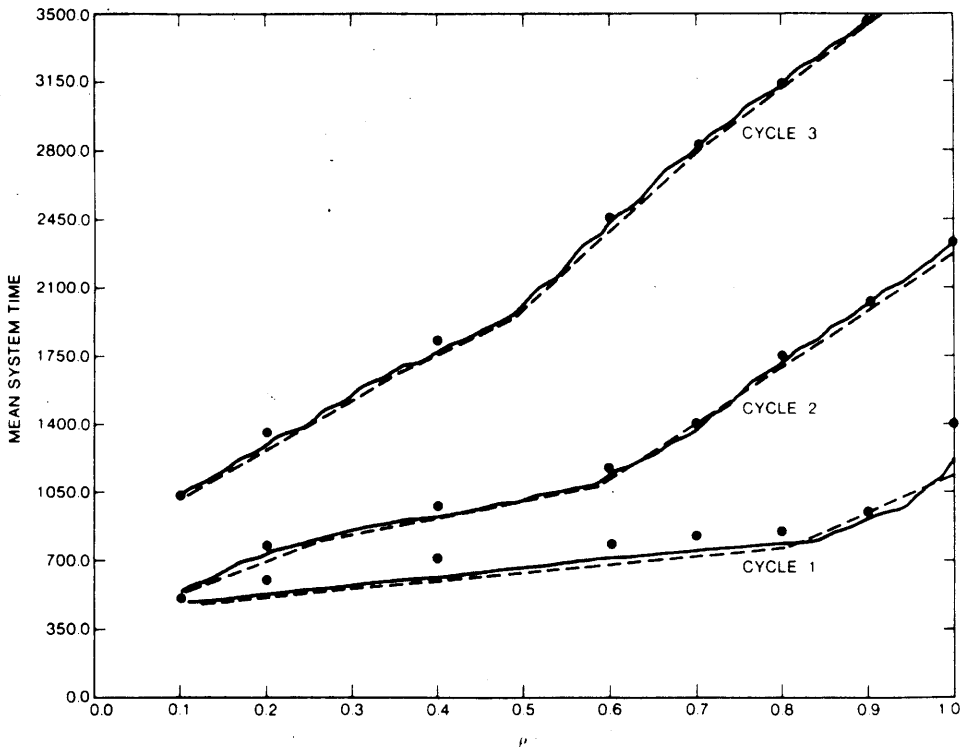


Fig. 7. Mean system time for the time frames of Fig. 6.

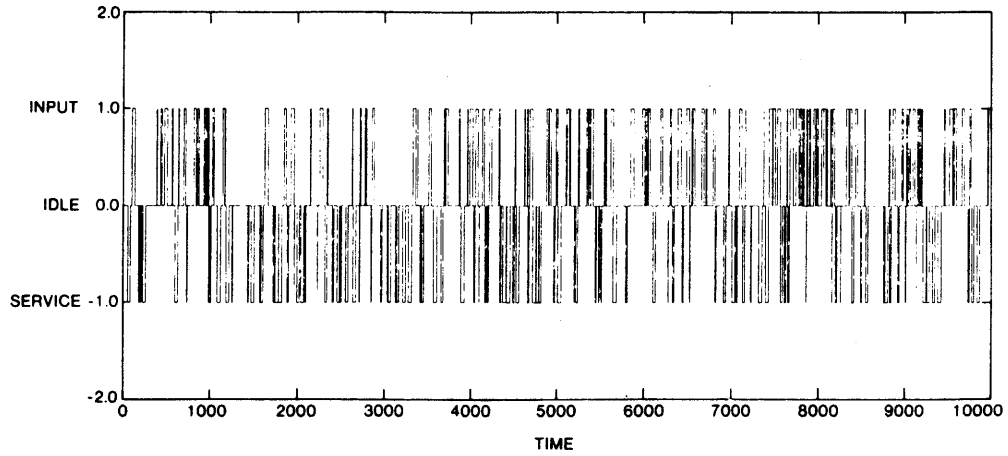


Fig. 8. A random time frame with reduced interval size.

arrivals over the interval (6000, 7800) must wait until the next service set of intervals (1200, 2300) to be processed. For lesser values of ρ, $\rho < 0.6$, these arrivals are serviced in the interval (7800, 9100) and, thus, suffer less delay. Naturally, as ρ increases, the proportion of messages that must wait until (1200, 2300) to be serviced grows, and so does the mean system delay. Each of the breaks in the piecewise linear approximation can be explained in this manner.

In Fig. 8 we have shown a frame for the same input parameters, but where the average size of the service and internal arrival intervals is equal to 20 time units instead of 200 as in those of Fig. 6. The corresponding mean system delay curve is shown in Fig. 9. We see a marked decrease in the mean system delay for this frame in comparision to the previous set of frames. This demonstrates the dependency of the mean system time upon the size of the intervals. If we adjust the frame to minimize the mean system delay, as shown in Fig. 10 (where for illustrative clarity we have only shown a portion of the frame), the resultant delay is approximately equal to 1 time unit throughout the entire range of ρ. For such a frame, the majority of the arrivals to the system are immediately serviced in the following service interval.

IV. The Capacity Assignment Problem

In the previous sections we determined, for a given frame, an approximation for the mean system delay packets' experience in passing through a node in a network using spatial TDMA. We observed that the ordering, size, and number of periods from the frame, allocated to internal arrival and service periods, can have a great influence on the mean system delay given in (1). In Section IV-A we formulate the capacity assignment problem for these networks and give a linear program to determine feasibility of the time vector t in Section IV-B. The capacity assignment program is very difficult to solve because the optimal capacity assignment depends upon the time vector t and the ordering of the slots assigned to the cliques of the clique cover. To obtain a computationally tractable approximate

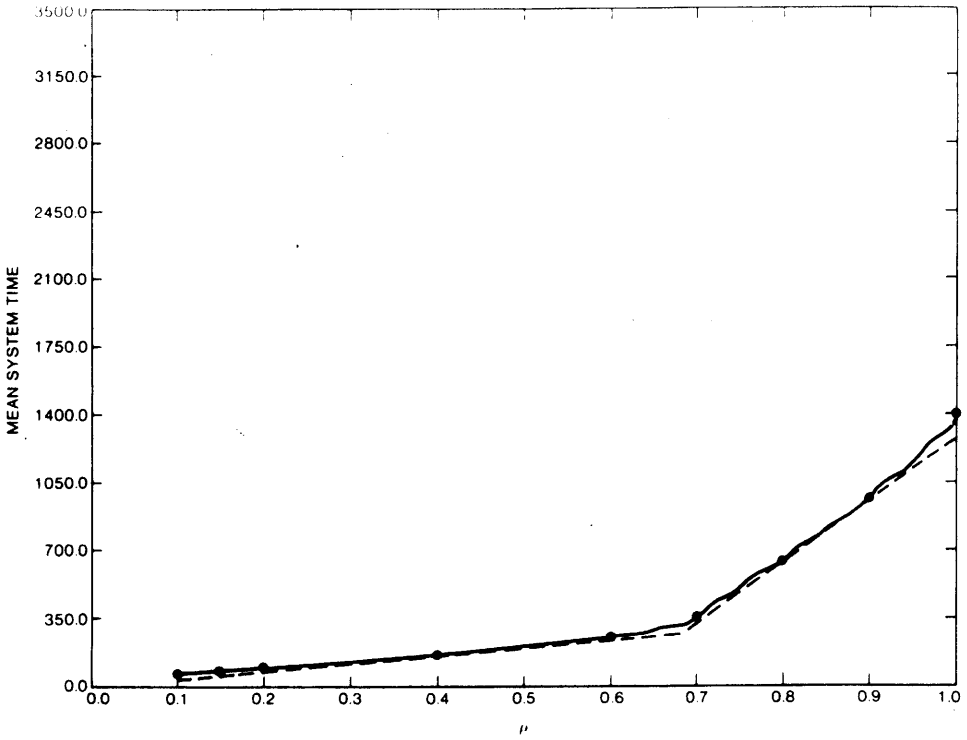


Fig. 9. Mean system time for the time frame of Fig. 8.

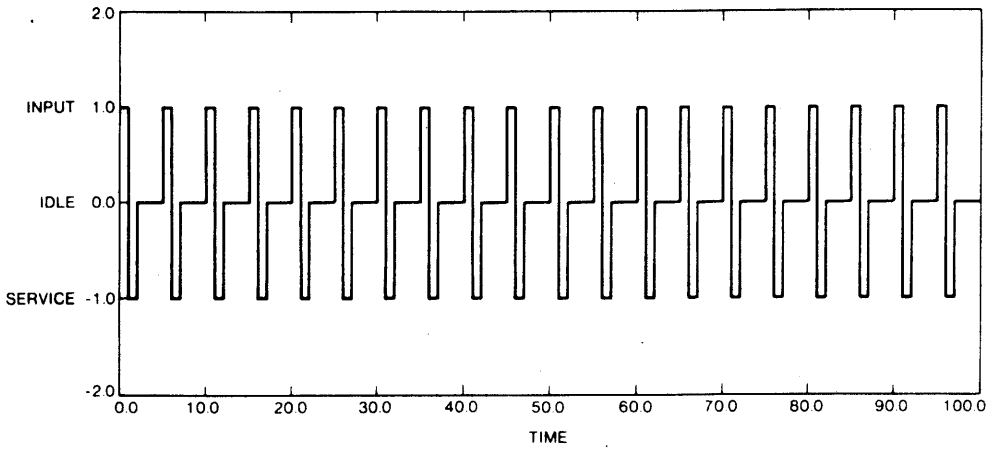


Fig. 10. An optimal time frame.

solution, we seek the optimal time vector t over a random assignment of the slots within the cycle T. Thus, in Section IV-B we develop an approximation to the mean system delay for frames in which the ordering of the slots has been randomly selected. This approximation is verified against simulation in Section IV-D.

For reasons of mathematical tractability, we will assume that elements of the time vector t are continuous rather than discrete variables. When the slot size is small in comparison to the cycle time (the usual case), we believe this will not produce significant errors.

A. Statement of Problem

We will assume that the total external traffic into the network is given by γ and that the expected flow of traffic between any two nodes i and j, $\gamma_{i,j}$, is known. A fixed route between all possible source–destination pairs is also assumed to be given. This allows us to calculate the average flow of traffic of class c through the ith node, which we denote as α_i^c, and also the total flow $\alpha_i = \Sigma_c \alpha_i^c$. We also assume that we are given a clique cover and denote the length of the cycle by T. Generalizing the notation of the previous section, we denote $T_{s,i}^c$ as the total time from the cycle during which node i transmits class c packets, and let $T_{\mathrm{id},i}$ and $T_{\mathrm{in},i}$ be the total time from the frame for idle and internal arrivals, respectively, at node i. Observe that these quantities are independent of the class of messages passing through node i, since a message of any class can arrive during any internal arrival period, and during idle periods neither receptions nor transmissions can occur for any class of packets. For a given clique cover and time vector t, one can easily calculate these time intervals. Let $\lambda_{\mathrm{ex},i}^c$ be the given external arrival rate (from the attached host) of class c packets from node i and let $\lambda_{\mathrm{in},i}^c$ be the corresponding internal arrival rate (from the network). By definition $\alpha_i^c = \lambda_{\mathrm{ex},i}^c T + \lambda_{\mathrm{in},i}^c T_{\mathrm{in},i}$. A time vector $t = (t_1, t_2, \cdots, t_k)$, where $t_i \geq 0$ is the time from the frame allocated to clique C_i, is said to be *feasible* if a) $T_{\mathrm{in},i} \geq \Sigma_c(\alpha_i^c - \lambda_{\mathrm{ex},i}^c T) \forall i$, b) $T_{s,i}^c \geq \alpha_i^c \forall i$, c, and c) $\|t\| \equiv \Sigma_{i=1}^k t_i = T$. For a given time vector t, the *ordering* of the frame is the particular permutation of the slots from the time vector to the cliques of the clique cover. For a given time vector there are $T!/\Pi_{i=1}^k t_i!$ possible orderings. For a given feasible time

vector t and queue i, let $D_i^c(t)$ be the mean system delay of class c messages passing through the queue at node i. Each $D_i^c(t)$ is to be calculated using the fluid approximation outlined previously in Section II and a randomization of slot allocations within a frame T which is discussed in Section IV-C.

With these assumptions we state the capacity assignment problem as

$$\underset{t \text{ and all possible orderings}}{\text{Minimize}} \sum_i \frac{\alpha_i}{\gamma} \left\{ \sum_c \frac{\alpha_i^c}{\alpha_i} D_i^c(t) \right\} \tag{3}$$

subject to

$$t_i \geq 0 \qquad \forall i$$

$$T - \|t\| = 0$$

$$T_{\text{in},i} - \sum_c (\alpha_i^c - \lambda_{\text{ex},i}^c T) \geq 0 \qquad \forall i$$

$$T_{s,i}^c - \alpha_i^c \geq 0 \qquad \forall i, c.$$

Before discussing the complexity of solving this problem, let us first establish that the feasibility region is convex. Suppose that t and t' are two feasible time vectors. To show convexity we need to show that $pt + (1 - p)t'$ where $0 \leq p \leq 1$ is also feasible. Let $T_{\text{in},i}(p)$ and $T_{s,i}^c(p)$ be the total amount of time of internal arrival and service intervals from the cycle for node i and class c for the time vector $pt + (1 - p)t'$. The convexity of the feasible region then follows, since both $T_{\text{in},i}(p)$ and $T_{s,i}^c(p)$ are either nondecreasing or nonincreasing in p. This implies that $T_{\text{in},i}(p)$ and $T_{s,i}^c(p)$ reach their minima at one of the end points (not necessarily the same point for all c), each of which correspond, by assumption, to a feasible time vector.

We should mention that one possible clique cover would consist of all the maximal cliques in the network, and that the optimization above would pick out "useless" cliques by assigning them a $t_i = 0$. There are, however, considerations that would not suggest this approach. For example, a designer of the network might prefer a particular clique cover (for example, to attempt to synchronize the flow of traffic for a particular source/destination pair), and also, increasing the size of the clique cover also increases the computational complexity of the program.

B. Feasibility of a Given Time Vector

In this section we establish a test for determining if a given clique cover, for a given set of queues, corresponding flows α_i, and having a cycle of T slots, permits a feasible time vector t. It is easy to create cases where, no matter how one adjusts the components of a time vector t, some nodal queues will have more flow into them than they can accommodate. We have already seen that the feasible region for program (3) is convex. This allows us then to formulate the feasibility problem as a linear program:

$$\underset{t}{\text{Minimize}} \ \|t\| \tag{4}$$

subject to

$$t_i \geq 0 \qquad \forall i$$

$$T_{\text{in},i} - \sum_c (\alpha_i^c - \lambda_{\text{ex},i}^c T) \geq 0 \qquad \forall i$$

$$T_{s,i}^c - \alpha_i^c \geq 0 \qquad \forall i, c.$$

If the solution to this linear program has $\|t\| \leq T$, any vector with $t + \epsilon$ where $\epsilon \geq 0$ and $\|\epsilon\| = T - \|t\|$ is a feasible time vector. Otherwise, no feasible time vector for a cycle length of T exists. We might add that an initial starting point for solving this linear program is to set all components of the time vector t equal to $t_j = \text{Max}_i\ \alpha_i T$, and that the program can halt as soon as the objective function $\|t\|$ becomes less than T.

C. Average Over Randomly Generated Frames

For a given ordering of slots within a frame, we have seen from the previous sections that a fluid approximation gave good results. Using this insight in this section, we will derive a closed form expression for the mean system delay averaged over a randomization of slot ordering within a cycle T. As in the queueing approximation section, we will concentrate on a single class of packets passing through a single node in the network, and suppress the subscript i and superscript c from our notations.

Suppose that for a given time frame, the number of slots allocated to idle, internal arrival, and service intervals is given by T_{id}, T_{in}, and T_s, respectively. Since we are assuming a randomly selected frame, we will generate the frame one slot at a time as outlined below. We will concentrate our discussion in terms of service slots, since similar statements can be made for the other kinds. Let $P_s(i)$ be the probability that the ith slot is a service slot. For the first slot we have $P_s(1) = T_s/T$. Suppose now that we have already generated $T' < T$ slots of the frame and let T_{id}', T_{in}', and T_{ex}' be the number of idle, internal, and service slots used in generating the frame up to this time. The next slot, $T' + 1$, will be drawn with a uniform probability from the remaining population of slots. At the $T' + 1$ step, the probability that the next slot selected is a service period is given by $P_s(T' + 1) = (T_s - T_s')/(T - T_s' - T_{\text{id}}' - T_{\text{in}}') \neq P_s(T' + 1)$. However, if T is large, one would expect that the probability, over most of the frame, of selecting a service period at each step does not differ much from $P_s(1)$. Intuitively T_s, T_{id}, and T_{in} decrease at a rate in proportion to their sizes, and since their sizes are assumed to be large, their relative proportions do not alter significantly over most of the frame. Only when most of the slots have been given out, i.e., when T' is close to T, will we expect a wide variation in P_s. The main point is that until this time, $P_s(T' + 1) \cong P_s(1)$ and the major component of the delay, as determined by the backlog curve of the fluid approximation, has already been determined. We thus conjecture that the delay obtained from generating the frame using the initial probability, T_s/T, throughout the frame generation process will not differ much from that which is generated by allowing $P_s(i)$ to change throughout frame generation.

Assume that the probability that the next slot is an idle, internal, or service slot is given by $P_{\text{id}} = T_{\text{id}}/T$, $P_{\text{in}} = T_{\text{in}}/T$, and $P_s = T_s/T$, respectively. We model backlog of packets at a nodal queue for a particular class of messages as a Markov chain where k, the backlog, is the state of the Markov chain. The state diagram for this Markov chain is shown in Fig. 11, where u_i, $i = 1, 2$, is defined to be the probability that the backlog increases by i in the next slot. Similarly, d is the probability that the backlog decreases by 1 in the next slot. We have

$$u_1 = P_{\text{id}}\lambda_{\text{ex}} + P_{\text{in}}[\lambda_{\text{ex}}(1 - \lambda_{\text{in}}) + \lambda_{\text{in}}(1 - \lambda_{\text{ex}})] \tag{5}$$

$$u_2 = P_{\text{in}}\lambda_{\text{in}}\lambda_{\text{ex}}$$

$$d = P_s(1 - \lambda_{\text{ex}})$$

and the evolution of the height of the backlog curve is given by a random walk with the above probabilities. The state

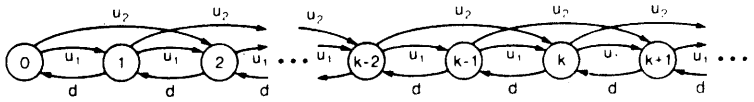


Fig. 11. State transitions for Markov chain.

transition equations are given by

$$(u_1+u_2)P_0 = dP_1 \tag{6}$$

$$(u_1+u_2+d)P_1 = u_1P_0 + dP_2$$

$$(u_1+u_2+d)P_k = u_2P_{k-2} + u_1P_{k-1} + dP_{k+1} \qquad k \geq 2$$

where P_k is the probability that the system is in state k. The approximation is generated in the same way as in the fluid approximation, namely, we find the area under the backlog curve and then divide by the average number of customers that enter the system. Suppose we knew the steady-state probabilities P_k. Points at a height k that move to $k+1$ incur an increase in the area that is equal to $k+0.5$, where the k arises from the rectangle of height k and width 1, and the 0.5 from the triangle having unit width and height. Similar calculations can be made for other steps and we can write the average area for each step as

$$A = P_0(u_2+0.5u_1) + \sum_{i=1}^{\infty} P_i[u_1(i+0.5) + u_2(i+1) + d(i-0.5)]$$

$$= P_0(u_2+0.5u_1) + \bar{P}(u_1+u_2+d)$$

$$+ (1-P_0)(u_2+0.5u_1-0.5d). \tag{7}$$

In these equations $\bar{P}$ is the average value of k. We can calculate both $\bar{P}$ and P_0 by using standard transform methods on equations (5). We find

$$\bar{P} = \frac{3u_2+u_1}{d-2u_2-u_1}$$

$$P_0 = \frac{d-2u_2-u_1}{d}.$$

Substituting this in (2) yields

$$A = \bar{P}(u_1+u_2+d). \tag{8}$$

Thus, we can calculate the mean system delay by dividing the average number of messages that arrive in a randomly selected slot. This equation is given below, where we explicitly represent it as the mean system delay for class c messages at the ith node:

$$D_i^c(t) = \frac{T}{(\lambda_{ex,i}{}^cT + \lambda_{in,i}{}^cT_{in,i})} \cdot \frac{(3u_{2,i}{}^c + u_{1,i}{}^c)}{(d_i{}^c - 2u_{2,i}{}^c - u_{1,i}{}^c)}(u_{1,i}{}^c + u_{2,i}{}^c + d_i{}^c) \tag{9}$$

for randomized slot allocations.

In the next section we will compare this formula to simulation results.

D. Comparing the Approximation with Randomly Generated Frames

In this section we show the results when (9) is compared to data obtained when we generate random frames, and calculate their mean system delay using the fluid approximation. Our procedure was, for a given set of parameter values λ_{ex}, λ_{in}, T_{id}, T_{in}, and T_s, to generate 1000 random frames and determine the minimum, maximum, mean, and variance of the delays for these samples. We then compared the mean system delay to (9) to determine if the approximation was close. We performed this procedure for many different selections of parameter values and all showed similar behavior. To demonstrate this behavior we have selected two sets to plot. Each of these sets had $T = 1000$. Set 1 has $T_{in} = 400$, $T_{ex} = 100$, $T_s = 500$, $\lambda_{ex} = 0.1$, and we varied λ_{in} over the range $0.02 \leq \lambda_{in} \leq 0.38$. Set 2 had $T_{in} = 300$, $T_{id} = 200$, $T_s = 500$, $\lambda_{in} = 0.2$, and λ_{ex} varied over $0.20 \leq \lambda_{ex} \leq 0.28$. In Fig. 12 we show how the approximation fared in relationship to the mean of the generated frames. We see in this figure that the approximation is very close to that of the sample mean, and that only for very large values of ρ does it break away from the generated frames. This is explained by the fact that the approximation is more stochastic than the randomly generated frames since it is given by a random walk, and that for high ρ values the random walk does not have a bound to its maximum height, in constrast to that of the generated frames which do.

We can extract more information about the randomly generated frames by looking at Table I.

In this table we list, for given values of ρ, for set 1, the mean system delay, variance, and minimum and maximum system delay that was found over the generated frames. We see that although the difference between the minimum and maximum system delay are often quite substantial (especially for large ρ), the variance is usually very small. This implies that the mean of all the generated frames is not much different from the mean of a particular generated frame. Another way to see this is to plot the coefficient of variation as a function of ρ. In Fig. 13 we plot this, as well as the variance, and see that the coefficient of variation is quite small throughout all ranges of ρ. Using this approximation, we are now in a position to formulate the capacity assignment problem. We can now state the capacity assignment for the case in which the frames have been randomly selected by substituting the value of $D_i^c(t)$ in (3) given by (9). Once again the feasibility region is convex, and it can also be shown by differentiating $D_i^c(\ )$ in (9) that the Hessian matrix of second partials is positive semidefinite, and thus, the objective function of (3), which is a convex combination of $D_i^c(t)$, is also convex. We thus have a convex programming problem which can be solved using any number of well-known solution techniques [7].

V. Conclusions

In this paper we have defined a channel access protocol that is assumed to operate for a packet radio environment in which the locations of the nodes of the network are assumed to be fixed and known. An approximation to the mean system delay of packets in the network was developed and compared to simulation, and a capacity assignment problem was formulated. The protocol has several desirable properties. Because it is collison-free, it simulates a traditional wire network. This suggests that one can adapt protocols specifically designed for wire networks, such as flow control and routing protocols, to operate in this radio environment. Unlike wire networks, the capacity of a channel can be algorithmically controlled, and thus, the network configuration can be altered to adapt to a changing network environment. One could imagine having several *capacity-plans* in

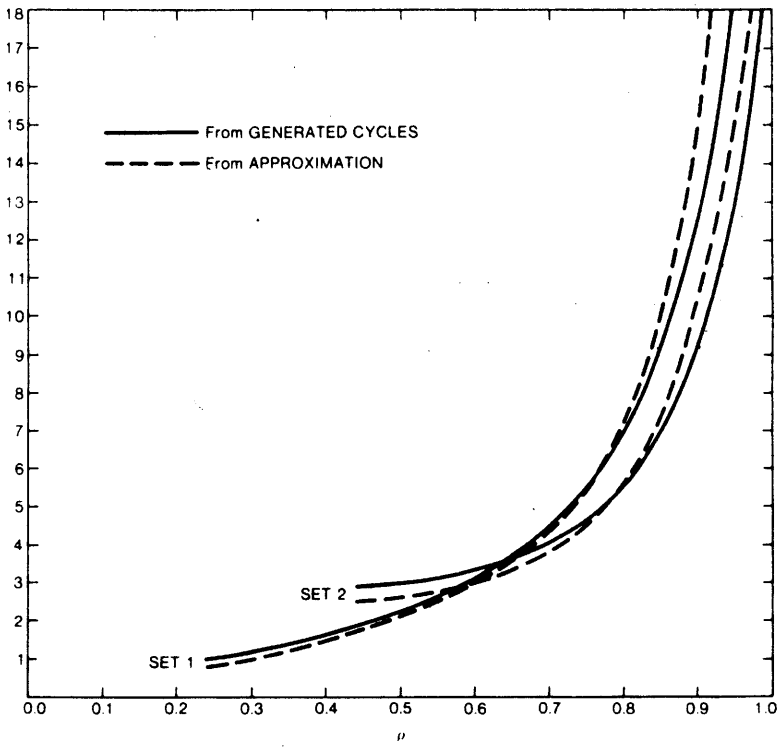


Fig. 12. Validating the approximation.

TABLE I

Utilization	Mean	Variance	Min Time	Max Time
0.240000	1.019933	0.059683	0.882035	1.292519
0.280000	1.173994	0.076922	1.000477	1.418889
0.320000	1.345148	0.097195	1.122492	1.737658
0.360000	1.519549	0.123856	1.228147	2.035303
0.400000	1.723441	0.145813	1.376908	2.425304
0.440000	1.919901	0.193245	1.435101	2.563087
0.480000	2.158198	0.218624	1.594046	3.138185
0.520000	2.434490	0.241395	1.902604	3.434316
0.560000	2.769812	0.325750	1.993773	4.291554
0.600000	3.172192	0.414819	2.390554	4.736672
0.640000	3.621928	0.522270	2.618372	5.937060
0.680000	4.222784	0.670217	2.910517	7.552518
0.720000	4.881158	0.869889	3.210058	9.927224
0.760000	5.741120	1.051270	3.566030	9.822580
0.800000	7.005569	1.534562	3.751551	15.805325
0.840000	8.766911	2.257944	5.079778	19.209290
0.880000	11.501342	3.460731	6.268044	32.473289
0.920000	15.094280	4.126865	7.613554	34.197002
0.960000	22.544538	7.200002	9.852835	65.090919

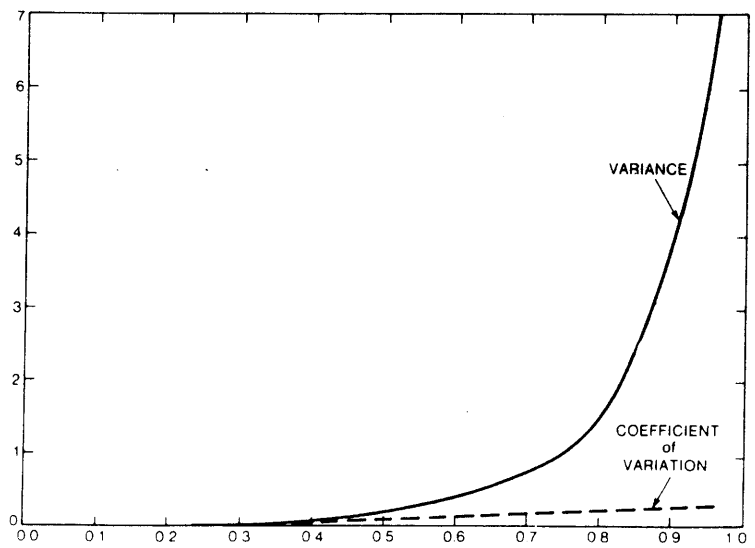


Fig. 13. Variance and coefficient of variations for randomly generated frames.

which the capacity of the links in the network was changed according to a schedule that mimics periodic alterations in network traffic. Adding a new node in such a network and readjusting the capacities in an optimal manner can be done by executing a program to determine the new set of maximal cliques and finding the optimal capacities. This could be useful in the context of a local area network that had frequent new subscribers.

There are many interesting extensions that are suggested by this work. It would be of interest to find the optimal ordering of the optimal time vector. This problem seems to be very difficult, but an efficient solution technique would be useful for extensions of this work. For example, if one prioritized traffic according to source-destination traffic, one would be faced with optimization problems similar to those encountered in traffic light control [8]. Under these conditions one would give priority to delays over certain paths, and would attempt to *synchronize* the placement of service slots so that messages arriving to a node would not have to wait long before being transmitted along the particular path. Naturally, synchronizing these slots along one path causes longer delays in other parts of the network in the same way traffic synchronization along one thoroughfare causes other paths to be unsynchronized. The optimization metric for this case would be a weighting, according to priority, of the delays along paths in the network.

Although the protocol is designed to operate in geographically static networks, a variation of it might prove to be useful in networks in which the nodes of the network move slowly with respect to their transmission ranges. In such a network, protocols would have to be executed to update the set of cliques and assignment of slots. Determining a practical protocol that is robust to network errors is an interesting research problem.

ACKNOWLEDGMENT

The authors would like to thank the referees for their insightful comments and criticisms which led to a greatly improved version of the paper.

REFERENCES

[1] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*. New York: McGraw-Hill, 1964; New York: Dover (reprint), 1972.

[2] D. G. Cantor and M. Gerla, "Capacity allocation in distributed computer networks," in *Proc. 7th Hawaii Int. Conf. Syst. Sci.*, Univ. Hawaii, Honolulu, Jan. 8-10, 1974, pp. 115-117.

[3] H. Frank, I. Frisch, W. Chou, and R. Van Slyke, "Optimal design of centralized computer networks," *Networks*, vol. 1, no. 1, pp. 43-57, 1971.

[4] B. Meister, H. Muller, and H. Rudin, "New optimization criteria for message-switching networks," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 256-260, June 1971.

[5] L. Kleinrock, *Queueing Systems, Vol. II, Computer Applications*. New York: Wiley-Interscience, 1976.

[6] J. Little, "A proof of the queueing formula $L = \lambda W$," *Oper. Res.*, vol. 9, pp. 383-387, Mar. 1961.

[7] M. Avriel, *Nonlinear Programming: Analysis and Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[8] K. Stoffers, "Scheduling of traffic lights—A new approach," *Transport. Res.*, vol. 2, pp. 199-234, 1968.

★

Randolph Nelson (S'79-M'82), for a biography, see p. 791 of the August 1985 issue of this TRANSACTIONS.

★

Leonard Kleinrock (S'55-M'64-SM'71-F'73), for a photograph and biography, see p. 638 of the July 1985 issue of this TRANSACTIONS.