INFORMATION SCIENCES 48, 253–287 (1989) 253

A Unified Minimum-Search Method for Resolving Contentions in Multiaccess Networks with Ternary Feedback*

JIE-YONG JUANG

Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60208-3118

and

BENJAMIN W. WAH

Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801

ABSTRACT

We propose a unified optimization method called *window protocol* for a class of CSMA and CSMA/CD protocols, which include the adaptive-tree-polling protocols, the urn protocols, the priority-access protocols, the arrival-time-window protocols, and the virtual-window protocol. Window protocols have the following features. First, each contending station in the network generates a contention parameter, and the channel will be allocated to one that generates the minimum. Second, a global window is maintained at every station to indicate the upper and lower bounds on the minimum. Lastly, a distributed window-control scheme is used to update the global window so that the minimum is uniquely isolated in the global window when the contention terminates. Based on this characterization, window protocols can be implemented by a distributed minimum-search procedure and optimized by a unique optimization method. The average overhead to resolve contentions for each transmitted packet is minimized by incorporating the channel load and the distribution of contention parameters into a dynamic programming formulation. Heuristic methods to set the windows and fast hardware implementation are also discussed.

*This research was partially supported by National Science Foundation Grant MIPS 85-19649 and Telecommunication Research Program of Northwestern University.

655 Avenue of the Americas, New York, NY 10010 0020-0255/89/$03.50

niversity.

nce Laboratory.

lass of CSMA and
urn protocols. the
-window protocol.
on in the network
that generates the
ate the upper and
is used to update
window when the
be implemented
mization method.
is minimized by
s into a dynamic
dware implemen-

on Grant MIPS
rsity.

-0255/89/$03.50

1. INTRODUCTION

In this paper we present a unified contention-resolution algorithm for the CSMA [22] and CSMA/CD [26] types of networks. These networks are characterized by a packet-switched random and multiple access channel. Every station in the network is able to sense the carrier status and obtain a consistent *ternary feedback* information representing the cases of idle channel, successful transmission, and collision. The time interval for detecting the collision of multiple transmissions is called a *contention slot*. No distinction has to be made between these two types of networks, at least within the scope of the contention-resolution problem.

A station in the network is *active* if it has a packet to transmit, and *idle* otherwise. Idle stations have no information on the status of others and may become active randomly. A *contention-resolution protocol* is used to resolve the contentions when the multiple stations wish to access the shared channel. An active station can either transmit its packet when the channel is free or defer its transmission to avoid collisions. Depending on the retransmission strategy, CSMA protocols can broadly be classified into nonadaptive and adaptive ones.

In a *nonadaptive CSMA protocol*, each active station has a fixed decision rule regardless of the channel load. Nonpersistent and p-persistent protocols are examples in this class [22]. It has been shown that an infinite-population CSMA network with a nonadaptive protocol is inherently unstable [11, 19, 8, 9]. The effective throughput of such a channel may approach zero due to possibly endless collisions.

In *adaptive protocols*, load information is used to make retransmission decisions. The load information may be estimated explicitly by a dedicated channel monitor as proposed by Kleinrock, and Yemini [23] or may be inferred implicitly from the contention experience of an individual station, as in the binary exponential backoff scheme of Ethernet [26]. A retransmission decision can be made either independently by individual stations or cooperatively among competing stations. In a cooperative contention-resolution algorithm, a subset of the active stations are enabled to transmit. The ternary-feedback collision-detection mechanism at each station determines whether there is no, one, or more than one active station in the enabled set. If the enabled set contains more than one or no active station, then the polling procedure is repeated until an active station is uniquely isolated in the enabled set, that is, when a successful transmission is detected. This approach has a higher throughput than noncooperative algorithms, but is more costly because all stations must monitor the channel activities continuously.

The efficiency of a contention-resolution algorithm depends on the proper choice of the set of enabled active stations to transmit in each contention slot.

To minimize the contention overhead, the subset polled must be small when the traffic is heavy and large otherwise. Extensive research has been conducted to determine the proper polling set. Protocols developed for networks of finite population include the urn protocols [23, 27], adaptive tree-polling protocols [4, 5, 6, 15, 17, 40], window protocols based on Markovian decision processes [18], and a polling scheme based on group testing [37, 3]. There are also many protocols developed for networks of infinite population. Among them are the arrival-time-window protocol proposed by Gallagher [12, 13] and subsequently improved by Mosely and Humblet [30, 31] to achieve the highest throughput so far. The same throughput was also achieved in the protocol developed by Tsybakov and his colleagues in the USSR [42, 41, 43]. Alternatively, various bounds on throughput of infinite-population networks were established to predict the maximum achievable throughput of a multiaccess channel [34, 10, 29, 16]. Note that the upper bound on throughput of a finite-population network is one, since a perfect scheduling can always be achieved under heavy traffic.

The major consideration in designing good contention-resolution protocols centers on the tradeoff between the amount of history information to be used in making retransmission decisions and the corresponding performance. It is desirable to have the best performance with only a tolerable amount of history information. When the entire history is modeled as a Markovian process, the throughput is maximized, but the real-time computational overhead is high. On the other hand, when little history information is used, the performance is load-dependent and is not satisfactory when the channel load is heavy.

In this paper, we focus on cooperative contention-resolution algorithms, in particular, the class of window protocols. We unify window protocols into a minimum-search procedure and optimize these protocols by a unique method. We have started out with the objective of minimizing the contention overhead for each packet sent so that this overhead is load-independent (not necessarily maximizing the throughput). By applying a common optimization algorithm and using identical inputs (information collected from broadcasts on the channel), each station is able to make the retransmission decision in a distributed fashion. The amount of dynamic history information needed is minimal and can usually be derived from information broadcast on the channel without any additional overhead.

In Section 2, the class of window protocols is characterized, and a distributed minimum-search procedure that provides a unified framework in this class is proposed. Optimizing this procedure by dynamic programming is also discussed. In Section 3, many protocols are shown to be members of the window protocols. These protocols are first briefly reviewed, and their mappings to the window protocols are then described. Section 4 shows the numerical evaluation of the proposed unified optimization scheme and the associated simulation results. Limitations of the proposed schemes are also discussed. Heuristic schemes and efficient hardware implementations are studied in Section 5.

2. WINDOW PROTOCOLS

2.1. STRUCTURE OF THE WINDOW PROTOCOLS

Window protocols discussed in this paper are characterized by the following properties. (1) Each active station is associated with a contention parameter that satisfies a complete linear ordering relation. (2) The channel is allocated to the station with the minimum contention parameter. (3) Stations enabled to transmit in each step of contention form a convex set in the contention-parameter domain. (4) The enabled set is determined in a distributed fashion and uniquely isolates the minimum when contentions terminate. To carry out the last step, each station is assumed to use information broadcast on the bus and apply an identical control algorithm to determine the enabled set. The resemblance of a convex set in a linear domain to an open window suggests the name *window protocols*. The domain of contention parameters can be either continuous or discrete, the distributions of contention parameters can be station-dependent or station-independent, and a discrete domain can be finite or infinite. Hlucyj's window protocols [18] are in a discrete finite domain, while Gallagher's window protocol [12] is in a continuous domain.

According to the characteristics outlined above, the contention-resolution algorithm of a window protocol is essentially a procedure to find the minimum among a set of distributed random numbers representing the contention parameters generated by the active stations. The minimum can be searched by the following search procedure. Suppose that the set of nonidentical contention parameters is $[x_1, \ldots, x_n]$ in the interval $(L, U]$, and $y_1 < y_2 < \cdots < y_n$ represent the sequence of x_i's sorted in ascending order. To search the minimum, an initial window is chosen with the lower bound at L and the upper bound at w_1 between L and U. There can be zero, one, or more than one y_i in this window. If there is exactly one number in this window, then it can be verified as the minimum, y_1. Otherwise, the window has to be updated. If the window contains no y_i, then the window has to be moved. The search is repeated with a window such that the lower bound is set at w_1 and the upper bound at w_2 between w_1 and U. In contrast, if the window contains more than one y_i, then it is reduced to a smaller size by choosing a new window with a lower bound at L and an upper bound between L and w_1.

In general, in the ith iteration, if the station containing the minimum has not been uniquely identified and the interval containing the minimum is $(L_i, U_i]$, then the window used in the ith iteration is $(L_i, w_i]$, where $L_i < w_i < U_i$. Stations with contention parameters inside the window are allowed to transmit. There are three possible outcomes. First, if a successful transmission is detected by the collision-detection mechanism, then the station containing the minimum is identified, and the contention-resolution process stops. Second, if the collision-detection mechanism detects no transmission, then $(L_{i+1}, U_{i+1}]$, the interval containing the minimum to be used in the $(i+1)$th iteration, is updated to be $(w_i, U_i]$. Lastly, if the collision-detection mechanism detects collision, then $(L_{i+1}, U_{i+1}]$ is updated to be $(L_i, w_i]$. Since the global window is updated in such a way that there is no contention parameter falling below its lower bound, the contention-resolution process is correct, and when it stops, the station containing the minimum is always identified.

Figure 1 illustrates the steps involved in the minimum-search procedure. Initially, five stations are active, and they sense that the channel is free. Each of them generates a random contention parameter in $(L, U]$, sets the window as $(L, w_1]$, and transmits in the next contention slot if its contention parameter falls in the window. Stations 3 and 5 are eliminated in the first iteration. As stations 1, 2, and 4 transmit, collision is detected. The stations reduce the upper bound of the interval to w_1 and set the window to $(L, w_2]$. The windows generated are identical for all stations, as they use identical window-control algorithms and observe the same channel feedback. In the second iteration, no transmission is detected because all contention parameters are outside the window. The lower bound of the interval is set at w_2, and all stations set the windows as $(w_2, w_3]$. In the third iteration, successful transmission is detected, and the process terminates.

2.2. *OPTIMIZATION OF WINDOW CONTROL*

A successful transmission in a CSMA network is preceded by a *contention period*. The length of a contention period is measured by the number of contention slots expended before the station containing the minimum contention parameter is identified. Since the duration of a contention slot is installation-dependent, minimizing the overhead in each contention period is equivalent to reducing the number of contention slots in a contention period. To this end, the window used in each step of the contention-resolution process must be chosen properly. If a large window is used, then the probability of collision is high. In contrast, a small window size increases the probability of no transmission. In either case, further contentions are needed.

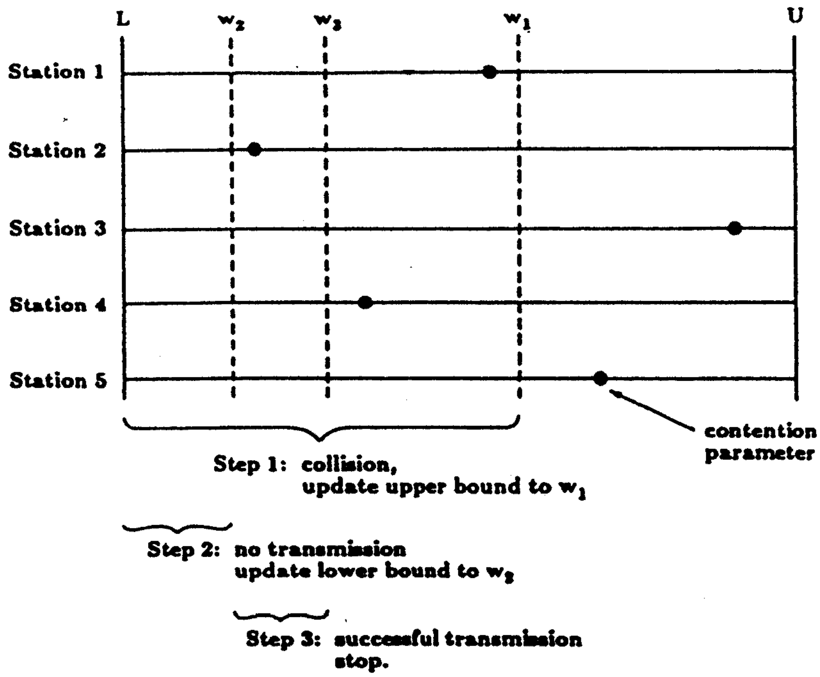


Fig. 1. An example to illustrate the updates of the global window to isolate the station with the minimum contention parameter (braces indicate windows used in different steps).

Various static and dynamic information can be employed in the window control. Generally, the more the amount of information is used, the higher is the throughput. The most commonly used information include the channel load and the way that the stations are organized. The fact that stations in a finite-population network can be organized into a tree structure leads to a higher possible throughput than that of an infinite-population network. On the other hand, dynamic information such as channel load is difficult to maintain in a network environment. A long-term average is used in most existing window protocols because it requires less frequent updates. However, there may be transient changes in channel load, and an optimization with respect to the average load may have an adverse effect on performance.

In this section, the optimal window control that minimizes the average number of contention slots is formulated into a recursive equation such that dynamic programming techniques can be applied to obtain the optimal sequence of windows. A window that minimizes the expected number of contention slots in a contention period depends not only on the probability of success in the current slot, but also on the number of future contention iterations in case that transmission is unsuccessful in the current slot. The following notation is first defined:

$C(a,b,n)$: minimum expected number of iterations to resolve the contention given that there are n contention parameters in $(a,U]$ and that collision occurs in the current window $(a,b]$;

$g(w,a,b,n)$: probability of *success* in the next iteration if a window of $(a,w]$, $a<w<b$, is used;

$l(w,a,b,n)$: probability of *collision* in the next iteration if a window of $(a,w]$, $a<w<b$, is used;

$r(w,a,b,n)$: probability of *no transmission* in the next iteration if a window of $(a,w]$, $a<w<b$, is used.

It follows directly from the above definitions that

$$l(w,a,b,n)+g(w,a,b,n)+r(w,a,b,n)=1. \tag{2.1}$$

As the principle of optimality is satisfied, the problem of minimizing the expected total number of iterations is reduced to that of finding w which minimizes the expected number of future iterations should collision or no transmission be detected in the current iteration. It can be formulated recursively as follows:

$$C(a,b,n)=\min_{a<w<b}\{1+0\cdot g(w,a,b,n)+C(a,w,n)\cdot l(w,a,b,n) + C(w,b,n)\cdot r(w,a,b,n)\}. \tag{2.2}$$

The probabilities $g(w,a,b,n)$, $l(w,a,b,n)$, and $r(w,a,b,n)$ can be derived from the distributions of contention parameters and the state of contention. When transmission experienced a collision, it can be determined that at least two of the x_i's lie in $(a,b]$ and that no x_i is smaller than a. This condition is designated as event A:

$$A=\{\text{at least two } x_i\text{'s are in } (a,b], \text{ given that all } x_i\text{'s are in } (a,U]\}.$$

Suppose that the window is reduced to $(a, w]$, $a < w < b$, in the next iteration. Three mutually exclusive events corresponding to the three possible outcomes of the contention process can be identified:

$$B = \{\text{exactly one of the } x_i\text{'s is in } (a, w], \text{ given that all } x_i\text{'s are in } (a, U]\};$$

$$C = \{\text{no } x_i \text{ is in } (a, w], \text{ given that all } x_i\text{'s are in } (a, U]\};$$

$$D = \{\text{more than one of the } x_i\text{'s are in } (a, w], \text{ given that all } x_i\text{'s are in } (a, U]\}.$$

From these events, the probabilities can be expressed as

$$g(w, a, b, n) = \Pr\{B \mid A\} = \frac{\Pr\{A \cap B\}}{\Pr\{A\}},$$

$$r(w, a, b, n) = \Pr\{C \mid A\} = \frac{\Pr\{A \cap C\}}{\Pr\{A\}}.$$

The set $A \cap B$ represents the event that exactly one of the x_i's is in $(a, w]$, that at least one x_i is in $(w, b]$, and that all others are in $(w, U]$. The set $A \cap C$ represents the event that at least two x_i's are in $(w, b]$, given that all x_i's are in $(w, U]$.

Let $F_i(x)$ $[f_i(x)]$ be the distribution [density] function that governs the generation of x_i, $1 \leqslant i \leqslant n$, where n is the number of contending stations. Event A occurs with probability

$$\Pr(A) = \frac{\prod_{i=1}^{n}[1 - F_i(a)] - \sum_{i=1}^{n}\left\{[F_i(b) - F_i(a)]\prod_{\substack{j=1 \\ j \neq i}}^{n}[1 - F_j(b)]\right\} - \prod_{i=1}^{n}[1 - F_i(b)]}{\prod_{i=1}^{n}[1 - F_i(a)]}. \tag{2.3}$$

The first and last terms of Equation (2.3) are the probabilities that all x_i's are greater than a and b, respectively. The second term is the probability that exactly one of the x_i's is in the window $(a, b]$. Similarly,

$$g(w,a,b,n) = \frac{\sum_{i=1}^{n}\left\{[F_i(w)-F_i(a)]\cdot\left[\sum_{\substack{j=1\\j\neq i}}^{n}[1-F_j(w)]-\prod_{\substack{j=1\\j\neq i}}^{n}[1-F_j(b)]\right]\right\}}{\Pr(A)\prod_{i=1}^{n}(1-F_i(a))}. \tag{2.4}$$

$$r(w,a,b,n) = \frac{\prod_{i=1}^{n}(1-F_i(w))-\sum_{i=1}^{n}\left[[F_i(b)-F_i(w)]\prod_{\substack{j=1\\j\neq i}}^{n}[1-F_j(b)]\right]-\prod_{i=1}^{n}[1-F_i(b)]}{\Pr(A)\prod_{i=1}^{n}[1-F_i(a)]}. \tag{2.5}$$

From Equations (2.1), (2.2), (2.4), and (2.5), an optimal window for the next contention slot can be derived once n, the channel load, and the $F_i(\cdot)$'s, the distributions of contention parameters, are known. Methods for estimating the channel load and distributions will be discussed in the next section.

The above optimization scheme minimizes each contention period independently. Therefore, it does not necessarily achieve the optimal throughput as in some schemes that keep track of every contention step [37, 3]. However, it has many advantages in a practical implementation: (1) It can adapt to instantaneous variations in channel load, since it is optimized with respect to the channel's instantaneous load rather than the average over a long period. (2) It utilizes the distributions of contention parameters to model the packet arrival processes; hence it is more general than schemes with the Poisson-arrival assumption. (3) It is independent of the type of distributions and hence can be applied to all window protocols. (4) It is a tractable combinatorial optimization problem instead of a complicated stochastic one.

Note that the optimization in Equation (2.2) depends on n, the instantaneous channel load, and not on N, the total number of stations in the network. Hence the model can be defined for an infinite-population network. However, when the distributions to generate the contention parameters are station-dependent, it would be very difficult for each station to know the n distribution functions used in the n active stations to generate the contention parameters without any global broadcast. For a realistic implementation, when the distributions are station-dependent, all active and idle stations in the network should participate in the contention-resolution process. In other words, the model can only be applied to a finite-population network.

To have a practical implementation, the protocol also requires that the distribution functions be sufficiently stable and time-invariant. Further, as not all transient parameters are considered in the optimization, the protocol is suboptimal and cannot maximize the channel throughput.

3. UNIFICATION OF EXISTING WINDOW PROTOCOLS

In this section, many notable CSMA (and CSMA/CD) protocols are shown to be members of the family of window protocols. The contention parameters that characterize these protocols are identified, and the distributions that govern the generation of contention parameters are derived. Methods for estimating the channel load are also presented. The computational efficiency of various methods will be discussed in the next section.

3.1. ADAPTIVE-TREE-POLLING PROTOCOLS

This series of CSMA protocols were developed for finite-population networks. In these protocols, stations are organized into a tree such that each leaf of the tree is associated with a station in the network. An enabled set to contend in a contention slot is identified as the leaves of a subtree that are determined by a preorder tree-traversal algorithm. The search begins at the root, and all active stations can transmit in the first contention slot. If a collision occurs, then the search continues recursively from the leftmost subtree to the rightmost subtree in the following contention slots. The search is successful when a single active station that is isolated in a subtree contends in a contention slot without collision. After the packet is transmitted, the next contention slot is reserved for the next subtree in the preorder search.

To achieve better performance, the size of initial enabled subtree should be chosen according to the channel load. It should be small if the channel load is high and large if the channel load is low. To this end, the nonterminal nodes of the tree may have different degrees under different traffic conditions. Capetanakis proposed a recursive scheme to construct a tree that maximized the channel efficiency with respect to Poisson arrivals [4–6]. The packet arrival rate was assumed to be known, and the search tree was reconstructed when the rate changed. Since subtrees are remembered for subsequent contention periods, the throughput of Capetanakis' protocol can be higher than our proposed protocol. However, his protocol does not allow new arrivals to contend immediately, while our proposed protocol does.

To transform an adaptive-tree-polling protocol into a window protocol, the leaves of the tree are first labeled from 1 to N, with the origin at 1. Every active station will generate a contention parameter equal to its label, while an idle station generates a large contention parameter, say $N+1$, which will never be enabled by the protocol. The objective in the current contention period is to find the station with the smallest contention parameter. This is equivalent to finding the active station with a contention parameter closest to the origin. After the current contention period, the labels of all stations are rotated so that the next station in the preorder search of the adaptive-tree-polling protocol becomes the new origin.

At any instant, the contention parameter generated by station i is either i or $N+1$; hence the distribution function that governs the generation can be formulated as follows:

$$F_i(k) = \begin{cases} 0 & k < i \\ \Pr\{\, i\text{th station is active}\} & i \leqslant k < N \\ 0 & k \geqslant N. \end{cases} \tag{3.1}$$

It is necessary to express this distribution function in terms of a measurable parameter indicating the channel load.

Let p be the probability that an idle station becomes active in a unit of time, and t_i be the elapsed time since the ith station was enabled to transmit previously. Then

$$\Pr\{\, i\text{th station is active} \mid p, t_i\} = 1-(1-p)^{t_i}. \tag{3.2}$$

The probability p may be expressed as a function of N, the total number of stations in the network, and n, the expected number of stations that will have transmitted when the window has circumscribed over all the stations once. The ration n/N is an indicator of the channel load, which is known to every station. Likewise, the t_i's are dynamic parameters, which should be known to all stations. If it takes θ_n units of time to transmit n packets, then the next station allowed to transmit must have been waiting for θ_n units of time. Hence the probability of a station becoming active after n packets have been transmitted is $1-(1-p)^{\theta_n}$. Denote the status of the ith station by a random number X_i whose value is one if the station is active and zero otherwise. By this definition,

$$\Pr(X_i = 1 \mid n) = 1-(1-p)^{\theta_n}. \tag{3.3}$$

It follows immediately that n may be expressed in terms of the X_i's as

$$n = E\left[\sum_{i=1}^{N} X_i\right] = N\Pr(X_i = 1 \mid n) = N\left[1-(1-p)^{\theta_n}\right], \tag{3.4}$$

where $E[\cdot]$ is the expected value. If the packet transmission time is fixed and equal to the collision-detection time, then $\theta_n \approx cn$, where c is the average number of contention iterations to resolve a contention. (In case that the collision-detection time is much smaller than a packet transmission time, then $\theta_n = n$.) Therefore, the following relation holds:

$$N\left[1-(1-p)^{cn}\right] = n. \tag{3.5}$$

Solving Equation (3.5) yields

$$p = 1-\left(1-\frac{n}{N}\right)^{1/cn}. \tag{3.6}$$

From Equations (3.1), (3.2), and (3.6), we obtain the distribution that governs the generation of contention parameters in the ith station as follows:

$$F_i(k) = \begin{cases} 0 & k < i, \\ 1-\left(1-\frac{n}{N}\right)^{t_i/cn} & i \leqslant k \leqslant N, \\ 1 & k > N. \end{cases} \tag{3.7}$$

The variables t_i and n in Equation (3.7) change as time evolves. To keep track of these variables, an N-bit register can be used at every station to record the status of all stations during the last circumscription of the window with respect to the current origin. The register is updated as the window moves from station to station. After a contention has been resolved, the station that won the contention is taken as the only active station among those enabled during this contention period. The bit corresponding to this active station is set to one, and those corresponding to idle ones are set to zeros in the enabled set. According to this updating scheme, n is the total number of bits set to one in the register, and the elapsed time of a station since last enabled is proportional to the number of bits set to one between this station and the current origin.

In the above scheme, in which all t_i's are computed with respect to the last circumscription of the window and the current origin, the real-time computational overhead to compute the windows in each contention slot is high. However, it is not feasible to compute the windows at design time and to look them up in real time, because there are 2^N combinations of possible states of the stations. Further, the merits of using only the state in the last circumscription of the window are questionable. To use the states of more than one circumscription of the window may not be practical. We propose a more efficient approximate scheme below.

In the approximate scheme, n is accumulated in a counter in each station, say i, as the long-term average of the number of packets transmitted between two points of time at which station i was enabled. The discrepancies in the values of n accumulated in different stations are small, since the long-term average is used. To compute t_i, it is assumed that the n packets transmitted in one circumscription of the window are *uniformly distributed* among the N stations. The probability that a packet was transmitted by one of the stations whose label is greater than i is $(N-i)/N$. The number of packets transmitted by this set of stations in one circumscription of the window has an incomplete binomial distribution. To compute the expected t_i, it is necessary to know the minimum and the maximum number of stations that could have transmitted when the window has moved from station i to the current origin. The maximum cannot exceed either n or $N-i$, and the minimum cannot be less than either 0 or $n-i$. Thus the probability that k out of n packets are transmitted by stations whose labels are greater than i is

$$p_k = \frac{\binom{n}{k}\left(\frac{N-i}{N}\right)^k\left(1-\frac{N-i}{N}\right)^{n-k}}{\sum_{j=\max(0,n-i)}^{\min(n,N-i)}\binom{n}{j}\left(\frac{N-i}{N}\right)^j\left(1-\frac{N-i}{N}\right)^{n-j}},$$

$$\max(0,n-i) \leqslant k \leqslant \min(n,N-i). \tag{3.8}$$

Substituting Equation (3.8) into Equation (3.7), the distribution function for station i, $1 \leqslant i \leqslant N$, to generate its contention parameter is

$$F_i(j) = \begin{cases} 0, & j < i \\ 1-\left(1-\frac{n}{N}\right)^{\left(\sum_{k=\max(0,n-i)}^{\min(n,N-i)} k p_k\right)/cn}, & i \leqslant j \leqslant N, \\ 1, & j > N. \end{cases} \tag{3.9}$$

In the approximate scheme, there are only $N+1$ possible values for n, and the sequence of windows used in each case can be computed at design time. This will be discussed in more detail in Sections 4 and 5.

3.2. URN PROTOCOL

The urn protocol proposed by Kleinrock and Yemini [23] and improved by Mittal and Venetsanopoulos [27] is another finite-population window protocol. It assumes that n out of N stations are active and applies the urn model in probability theory to find the best estimate of the number of stations to be enabled such that the probability of having exactly one active station in the enabled set is maximized. An enabled set is further divided into two equal subsets if it contains more than one active station. In applying the urn model, active stations are assumed to be uniformly distributed over the stations. The uniform distribution can be maintained by having a synchronized pseudorandom-number generator at each station, and the stations to be enabled are determined by a random sequence. The major problem in this approach is the synchronization of multiple random-number generators.

Alternatively, a rotating-window approach was proposed in which the initial window size was determined by the urn model. This approach is similar to the adaptive-tree-polling protocol in the sense that the stations are enabled sequentially in the spatial domain. However, due to the round-robin service discipline, those stations closer to the origin of the window have a longer elapsed time since last enabled, and thus a higher probability of becoming active. Consequently, the active stations are no longer uniformly distributed over the space of stations, and the urn model may fail to work properly. Moreover, the urn protocol has to be supported by a subchannel to estimate n, which is not viable in practice.

It is easy to show that an urn protocol can be transformed into a window protocol in exactly the same way in which the adaptive-tree-polling protocol is transformed (as described in last section).

3.3. *PRIORITY-CSMA PROTOCOLS*

Many CSMA protocols for handling priority messages have been suggested in recent years [7, 35, 38, 14, 32, 36]. They can be classified into linear protocols and logarithmic protocols. In a *linear protocol*, a slot is reserved for each priority level during the resolution of priorities. An active station contends during the slot reserved for its highest local priority level. The process stops when the highest global priority level is determined, which may be followed by another contention phase to select one active station in this priority level. This scheme is good when high-priority messages are predominantly sent. In contrast, a *logarithmic protocol* can resolve the highest priority level in $O(\log_2 P)$ contention slots by using a binary-divide scheme, where P is the maximum number of priority levels [32]. This assumes that the highest global priority level is equally likely to be any one of the P priority levels. Neither of these schemes is able to adapt to transient traffic patterns.

Since there may be more than one station with messages belonging to the highest priority, the entire contention-resolution period can be divided into two phases. In the first phase, the highest priority level is identified, while in the second phase, one of the stations in this level is chosen. Another way is to first transform q_i, the highest priority level in station i, into a contention parameter $x(q_i)$ such that

$$x(q) = P - q - \gamma, \tag{3.10}$$

where γ is a random number uniformly distributed in (0,1]. γ is introduced in the transformation to break ties among active stations with the highest-priority message. This transformed contention parameter has a piecewise continuous distribution. The objective is to use the window protocol to identify the station with the minimum transformed contention parameter.

To derive the distributions of contention parameters, it is assumed that the message arrivals of priority class i at station j is Poisson distributed with rate $\lambda_{i,j}$, and that the corresponding service time is exponentially distributed with rate $\mu_{i,j}$. Let λ_i be $\Sigma_{j=1}^{N}\lambda_{i,j}$. Suppose that the channel can serve a message of class i at rate μ_i. Since a lower-priority message can only be served during an idle period between serving higher-priority messages, the effective service rate of class-i messages in the channel is

$$\mu_i^e = \mu \prod_{k=i+1}^{P} (1-\rho_k) \quad \text{for } i=1,\dots,P-1, \quad \text{and} \quad \mu_P^e = \mu, \tag{3.11}$$

where ρ_k is the traffic intensity of class-k messages in the system. By definition,

$$\rho_i, = \frac{\lambda_i}{\mu_i^e}, \qquad i=1,\dots,P. \tag{3.12}$$

Since stations have equal access to the channel, the effective service rate for class-i messages at station j is

$$\mu_{i,j}^e = \frac{\mu_i^e}{N}, \qquad i=1,\dots,P, \quad j=1,\dots,N. \tag{3.13}$$

The traffic intensity of class-i messages at station j is

$$\rho_{i,j} = \frac{\lambda_{i,j}}{\mu_{i,j}^e}, \qquad i=1,\dots,P, \quad j=1,\dots,N. \tag{3.14}$$

Class-i messages will be empty at station j with probability $1-\rho_{i,j}$. Thus station j will generate a contention parameter of value q with probability

$$f_j(q) = \rho_{q,j} \prod_{i=q+1}^{P} (1-\rho_{i,j}), \qquad j=1,\dots,N, \quad q=1,\dots,P. \tag{3.15}$$

The distribution that governs the generation of the contention parameters at station j can be obtained by applying the transformation in Equation (3.10) to the following distribution:

$$F_j(q) = \sum_{k=1}^{q} f_j(k), \qquad j=1,\ldots,N, \quad q=1,\ldots,P. \tag{3.16}$$

To estimate the $\lambda_{i,j}$'s, the waiting time of a message can be broadcast after the message has been transmitted. This information allows the average waiting time of a message in a class to be estimated. Denote the average waiting time of class-i packets transmitted by station j as $T_{i,j}$. Then

$$T_{i,j} = \frac{1}{\mu_{i,j}(1-\rho_{i,j})}. \tag{3.17}$$

Rearranging Equation (3.17) yields

$$\lambda_{i,j} = \mu_{i,j} - \frac{1}{T_{i,j}}. \tag{3.18}$$

It is not difficult to show that the all stations have identical information in optimizing the windows used. Moreover, all stations participate in identifying the station with the highest-priority message; hence $n = N$.

3.4. *ARRIVAL-TIME-WINDOW PROTOCOL*

Gallagher proposed a window protocol on the time axis in which all stations have a common window of length u in the past [12, 39]. Stations with packets arriving during the window are allowed to transmit. If there is no transmission or a successful transmission, then the window is advanced to the beginning of unresolved interval; otherwise, the window is reduced to a fraction f of its original size, and the process is repeated until a packet is transmitted successfully. The parameters u and f are chosen to optimize the performance with respect to the packet arrival rate. A binary-divide scheme was used in Gallagher's protocol. Subsequently, Mosely and Humblet applied the nondiscounted Markov decision process to refine the protocol and achieved an even higher channel throughput [30, 31]. Towsley relaxed the constraint on collision detection and assumed that the exact number of stations involved in a collision is known. A recursive procedure was used to maximize the channel efficiency, but little improvement has been obtained [39]. Kurose, Schwartz, and Yemini successfully applied the protocol to time-constrained applications [24, 25].

It is easy to map this protocol to a window protocol, since the elapsed time from the origin of the window to the instant of packet arrival may serve as a contention parameter. The earliest arrived packet is associated with the station having the minimum contention parameter. The distributions which govern the generation of contention parameters can be derived as follows. Suppose that the window begins at time O, that the current time is T, and that the arrival process at each station is Poisson. The distribution of the arrival time for the earliest arrived packet in station i conditioned on T and O is

$$F_i(t \mid O < t \leqslant T) = \begin{cases} 0, & t \leqslant O, \\ \dfrac{1 - e^{-\lambda_i(t-O)}}{1 - e^{-\lambda_i(T-O)}}, & O < t < T, \quad i = 1, \ldots, N, \\ 1, & t \geqslant T, \end{cases} \tag{3.19}$$

where λ_i is the packet arrival rate at station i. Note that if $\lambda_i \neq \lambda_j$, then $F_i(t) \neq F_j(t)$.

3.5. VIRTUAL-WINDOW PROTOCOL

A virtual-window protocol was proposed by Wah and Juang to support distributed-processing applications such as resource scheduling [21, 46], load balancing [44, 2], and local distributed databases [47]. It can also be applied to resolve contentions in CSMA/CD networks [20, 45, 46]. Each of the n active stations generates a random number from the uniform distribution $U(0,1)$ as its contention parameter. That is,

$$F_i(y) = \begin{cases} 0, & y \leqslant 0 \\ y, & 0 < y \leqslant 1, \\ 1, & y > 1 \end{cases} \qquad i = 1, \ldots, N. \tag{3.20}$$

A contention parameter in this protocol is only a dummy argument. It may be either generated randomly or obtained by a transformation from an application-dependent parameter.

In the virtual-window protocol, the channel load can be estimated easily. After the tth message has been transmitted, the window $(L, w(t)]$ that successfully isolate the station with the minimum contention parameter is known to all stations. A maximum-likelihood estimate of $n(t)$, the number of stations that have participated in the contention, can be computed from a likelihood function on the probability that the minimum lies in $(L, w(t)]$. Assuming that the contention parameters are independently and uniformly distributed in $(0,1]$, the likelihood function is derived as

$$\mathrm{LK}(0, w(t), \hat{n}(t)) = \Pr(0 < Y_1 \leqslant w(t) < Y_2) = \hat{n}(t) w(t) [1 - w(t)]^{\hat{n}(t)-1} \tag{3.21}$$

where $\hat{n}(t)$ is the maximum-likelihood estimate of $n(t)$, and Y_i is the random variable for the ith minimum. LK$(0, w(t), \hat{n}(t))$ is maximized at

$$\hat{n}(t) = \left\lceil \frac{-1}{\log_e[1 - w(t)]} \right\rceil, \qquad 0 < w(t) < 1. \tag{3.22}$$

The number of stations contending to transmit the $(t+1)$th message can be obtained by adding to $\hat{n}(t)$ the expected arrivals after the tth message has been transmitted.

The accuracy of estimation can be improved by using information on previous windows that successfully isolate a single station. A technique in time-series analysis called the autoregressive-moving-average (ARMA) model can be applied to obtain an estimated window based on all previous windows, $w(1), w(2), \ldots, w(t)$. A simple example is to compute a moving average $w_{\mathrm{mv}}(t)$ using the following formula.

$$w_{\mathrm{mv}}(t) = \sum_{i=1}^{t} \frac{w(i)}{2^{t-i}}. \tag{3.23}$$

The above equation can be rewritten into a recursive expression as follows:

$$w_{\mathrm{mv}}(t) = \frac{w_{\mathrm{mv}}(t-1) + w(t)}{2}. \tag{3.24}$$

Hence $w_{\mathrm{mv}}(t)$ can be obtained by taking the average of the current window and the moving average in the previous contention phase. The value of $w_{\mathrm{mv}}(t)$ is then used in Equation (3.22) to estimate the channel load. This load-estimation method can also be applied to other window protocols with identical continuous distributions.

4. NUMERICAL EVALUATIONS AND SIMULATIONS

In this section, the dynamic-programming optimization method is evaluated with respect to identical and nonidentical distributions. Both continuous and discrete distributions are considered in each case. The average numbers of iterations to resolve contentions among active stations were first obtained by evaluating Equation (2.2) numerically. The values obtained were then compared against simulation results. The simulator associated with each window protocol was coded in F77, and run on a DEC VAX 11/780 computer. In the simulator, a station was represented by a random-number generator that generated a contention parameter if the station was active, and a collision-detection mechanism was modeled by a counter which counts the number of contention parameters in a given window. Each result was obtained by simulating a number of times with different seeds until a 95% confidence interval of less than 0.2 was obtained. In each run, more than 10,000 packets were simulated.

4.1. IDENTICAL DISTRIBUTIONS

It is well known that a continuous distribution can be transformed into a uniform distribution [33]. x_i, a random variable generated by a continuous distribution $F_i(\cdot)$, can be transformed into another random variable uniformly distributed over (0,1] by replacing x_i with

$$x_i' = F_i(x_i). \tag{4.1}$$

Since this transformation is a one-to-one mapping, if the distributions to generate the contention parameters are identical and continuous, the optimization performed on the transformed contention parameters can be shown to be equivalent to the original optimization. Similarly, identical discrete distributions can be transformed into a continuous uniform distribution using Equations (4.1) and (3.10). Hence all protocols with identical distributions discussed in the last section can be optimized in a similar way, and Equation (2.2) has only to be evaluated with respect to the case in which all contention parameters are uniformly distributed in (0,1].

Note that transforming a discrete distribution to a continuous one tends to increase the uncertainty of finding the minimum. Although it is simple to optimize a protocol of discrete identical distributions using such a transformation, the performance of the resulting protocol will be degraded. It is better to handle the case with discrete identical distributions using the schemes to be discussed later.

Another advantage when the distributions are identical is that an infinite-population network can be assumed. The optimization in Equation (2.2) only requires the current number of active stations contending for the channel to be estimated [Equation (3.22)] rather than the total number of stations in the network. Moreover, the estimate on channel load provides an additional measure on the transient behavior, which cannot be derived from statistical measures in the distribution functions.

With identical continuous distributions, Equations (2.4) and (2.5) can be reduced to simpler forms as shown below:

$$g(w,a,b,n) = \frac{n(w-a)\left[(1-w)^{n-1}-(1-b)^{n-1}\right]}{(1-a)^n-(1-b)^n-n(b-a)(1-b)^{n-1}}, \tag{4.2}$$

$$r(w,a,b,n) = \frac{(1-w)^n-(1-b)^n-n(b-w)(1-b)^{n-1}}{(1-a)^n-(1-b)^n-n(b-a)(1-b)^{n-1}}. \tag{4.3}$$

TABLE 1
Numerical Evaluation and Simulation Results of Infinite-Population Window Protocols under Continuous Dynamic Programming Window Control

Number of contending stations, n	Numerical evaluations with truncation when $b-a<1/(10n)$	Simulations with binary divide when $b-a<1/(10n)$
5	2.21	2.33
10	2.29	2.42
15	2.32	2.52
20	2.33	2.43
25	2.34	2.47
30	2.34	2.56
35	2.35	2.57
40	2.35	2.48

In a continuous domain, there are infinite numbers in the interval $(a,b]$. Accordingly, there are infinite values of $C(a,b,n)$ to be computed, each of which involves infinite levels of recursion. As a result, exact numerical evaluation of Equation (2.2) in a continuous domain is impossible. Some boundary conditions, must be set to truncate the evaluation process after a number of levels of recursion. In our evaluations, the truncation was done by setting $C(a,b)=1$ if $b-a<\delta$, where δ is a small positive number. The truncation implies that contentions can be resolved in one step if the enabled set is sufficiently small. This is justified by the small probability of having more than one contention parameter in such a small interval. To maintain the same level of accuracy, δ must be very small when n is large and should be relatively large if n is small. It was set to the reciprocal of ten times the number of contending stations in our evaluations. Note that the resulting dynamic-programming tree is a skewed tree. The numerical evaluations in Table 1 show that the average number of contention iterations is consistently smaller than 2.4 and increases very slowly with respect to n.

It is possible to have more than one contention parameters in a window smaller than δ, but no further refined windows were supplied by dynamic programming in our simulations, and the binary-divide rule was used to divide the window into equal halves. The results obtained are again summarized in Table 1. The difference between the simulation results and numerical evaluations is about 5%, which is due to the effect of truncations of recursions in numerical evaluations and the use of binary-divide rule in simulations.

To further investigate the effects of truncation, numerical evaluations and simulations were conducted on different values of δ using the formula $\delta=1/rn$. The results in Table 2 indicate that the truncation effect becomes smaller as r

TABLE 2

The Effects of Truncation on the Performance of Infinite-Population Window Protocols under Continuous Dynamic Programming Window Control [$n = 20$; $\delta = 1/(rn)$]

r	Numerical evaluations with truncation when $b - a < 1/(rn)$	Simulations with binary-divide when $b - a < 1/(rn)$
1	1.54	2.48
2	1.96	2.49
3	2.12	2.49
4	2.19	2.52
5	2.24	2.48
6	2.27	2.56
7	2.29	2.52
8	2.30	2.50
9	2.32	2.45
10	2.33	2.47
11	2.34	2.50
12	2.35	2.55
13	2.35	2.47
14	2.35	2.48
15	2.36	2.42
16	2.36	2.52
17	2.36	2.50
18	2.37	2.51
19	2.37	2.48
20	2.37	2.45

increases. For small r, errors due to truncation are substantial. This is expected, because a window smaller than δ has a high probability of not being resolved, while this was resolved by the suboptimal binary-divide rule in the simulations. When r is sufficiently large, say larger than or equal to ten, errors due to truncation is insignificant. The simulation results in Table 2 show that the performance is quite independent of r. In this case, even if the window is unresolved when it is smaller than δ, it is likely that there are only two remaining stations in this small window, and the binary-divide rule is a good heuristic window-control rule to resolve the collisions [21].

Both numerical evaluations and simulation results suggest that the contention overhead of a continuous-window protocol with dynamic-programming window control is independent of the channel load and is bounded asymptotically. Arrow et al. had studied a similar problem with the difference that the number of contending stations in a collided window was assumed to be known exactly [1]. The problem was formulated as a finite recursion, and an asymptotic bound of 2.4 iterations was obtained by numerical evaluations. We have obtained comparable results when only ternary information on collision is

ulation Window Protocols
ol [$n = 20$; $\delta = 1/(rn)$]

Simulations with binary-divide when $b-a<1/(rn)$
2.48
2.49
2.49
2.52
2.48
2.56
2.52
2.50
2.45
2.47
2.50
2.55
2.47
2.48
2.42
2.52
2.50
2.51
2.48
2.45

stantial. This is expected.
ity of not being resolved,
de rule in the simulations.
ual to ten, errors due to
n Table 2 show that the
e, even if the window is
that there are only two
ary-divide rule is a good
[21].
ts suggest that the con-
th dynamic-programming
nd is bounded asymptoti-
th the difference that the
vas assumed to be known
ecursion, and an asymp-
ical evaluations. We have
ormation on collision is

available and the infinite dynamic-programming tree is truncated. This shows that the information on the exact number of contending stations is insignificant.

4.2. NONIDENTICAL DISTRIBUTIONS

If the distributions are nonidentical, then the evaluation of the dynamic-programming formulation is complex. One method is to first transform each nonidentical distribution into a uniform one using Equation (4.1). However, the linear-ordering property is lost after the transformation. For example, in the arrival-time-window protocol with different packet arrival rates at each station, a different transformation will be used in each station. As a result, the original order of packet arrivals may not be preserved after the transformation because a packet arriving earlier at a light-traffic station may be transformed into a larger contention parameter than a packet arriving later at a heavy-traffic station. Due to this phenomenon, the first-come-first-served discipline with nonidentical arrival rates studied by Gallagher and others [12, 24, 25] cannot be implemented using transformed contention parameters. Similarly, the order of priorities in priority-CSMA protocols may be changed by transformations if the distributions for generating priority levels at various stations are nonidentical.

Another method is to optimize the windows directly from the distribution functions. There are two problems associated with this approach. First, when the distributions are nonidentical, it is difficult for each station to know the distribution functions used in the corresponding active stations without sequentially polling all active stations. This will result in an intolerable network overhead. To avoid information from the active stations, all active and idle stations should generate contention parameters and contend for the bus. Consequently, only a finite-population network model can be assumed, and the transient-load behavior is lost. Second, and more seriously, the complexity of the optimization problem using nonidentical distributions is increased significantly. For each distribution, there may be one or more parameters indicating the dynamic transient behavior. For example, in the adaptive-tree-polling protocol, the parameter t_i represents the dynamic elapsed time for station i since last enabled. In the priority-CSMA protocol, $T_{i,j}$ represents the station-dependent waiting time of class-i packets. In the arrival-time-window protocol, λ_i is the station-dependent arrival rate of station i. The precomputation of the dynamic-programming tree at design time for all possible combinations of transient parameters would result in a very large table. The real-time search of the dynamic-programming solution is also infeasible due to its computational overhead. A viable approach is to use the average of these transient parameters and to compute the dynamic-programming solution for a much smaller set of combinations of parameters. The dynamic transient behavior of systems is again lost. An example of this approach is shown in the estimation of t_i in Equation (3.9) with a probabilistic model.

The discrete distributions derived for both tree-polling and urn protocols are used in our evaluations. A direct comparison between our proposed method and existing tree-polling and urn protocols is not made, because information on the arrival process is not assumed in our model.

Without loss of generality, we assume that the origin of the window is at station 1. There are two properties of these distributions that contribute to the reduction of complexities in Equations (2.3) through (2.5): (i) $F_i(k) = 0$ for $k < i$, and (ii) $F_i(a) = F_i(b) = 1$ for $i \leqslant a, b \leqslant N$. Hence

$$\Pr(A) = \frac{\prod_{i=1}^{a}[1 - F_i(a)] - \sum_{i=a}^{b}\left\{F_i(b)\prod_{\substack{j=1\\ j\neq i}}^{b}[1 - F_j(b)]\right\} - \prod_{i=1}^{b}(1 - F_i(b))}{\prod_{i=1}^{a-1}[1 - F_i(a)]}. \tag{4.4}$$

$$g(w, a, b, N) = \frac{\sum_{i=a}^{w}\left\{F_i(w)\cdot\left[\prod_{\substack{j=1\\ j\neq i}}^{w}[1 - F_j(w)] - \prod_{\substack{j=1\\ j\neq i}}^{b}[1 - F_j(b)]\right]\right\}}{\Pr(A)\prod_{i=1}^{a-1}[1 - F_i(a)]}. \tag{4.5}$$

$$r(w, a, b, N) = \frac{\prod_{i=1}^{w}[1 - F_i(w)] - \sum_{i=w}^{b}\left[F_i(b)\prod_{\substack{j=1\\ j\neq i}}^{b}[1 - F_j(b)]\right] - \prod_{i=1}^{b}[1 - F_i(b)]}{\Pr(A)\prod_{i=1}^{a-1}[1 - F_i(a)]}. \tag{4.6}$$

Unlike the continuous case, there are only finitely many possible candidates in each recursion, and the recursion procedure terminates in a finite number of steps in discrete domains. Thus, Equation (2.2) can be evaluated exactly for given discrete distributions. In our evaluations, the estimated distribution in Equation (3.9) was computed with respect to different values of n and N. For each distribution obtained, we compute the average number of contention iterations using Equation (2.2). The results obtained are shown as bold lines in Figure 2 and illustrate that discrete window protocols outperform a continuous one. In most cases, contention can be resolved in one step, and near-perfect scheduling can be achieved when the channel load is heavy.

nsient behavior of systems is
wn in the estimation of t_i in

e-polling and urn protocols are
between our proposed method
made, because information on

he origin of the window is at
ributions that contribute to the
rough (2.5): (i) $F_i(k) = 0$ for
Hence

$$F_j(b)]\Big\} - \prod_{i=1}^{b}(1 - F_j(b)) \quad (4.4)$$

$$)]$$

$$[1 - F_j(b)]\Big]\Big\}. \quad (4.5)$$

$$F_j(b)]\Big] - \prod_{i=1}^{b}[1 - F_i(b)] \quad (4.6)$$

$$F_i(a)]$$

nitely many possible candidates
terminates in a finite number of
2) can be evaluated exactly for
s, the estimated distribution in
ifferent values of n and N. For
average number of contention
ained are shown as bold lines in
otocols outperform a continuous
d in one step, and near-perfect
oad is heavy.

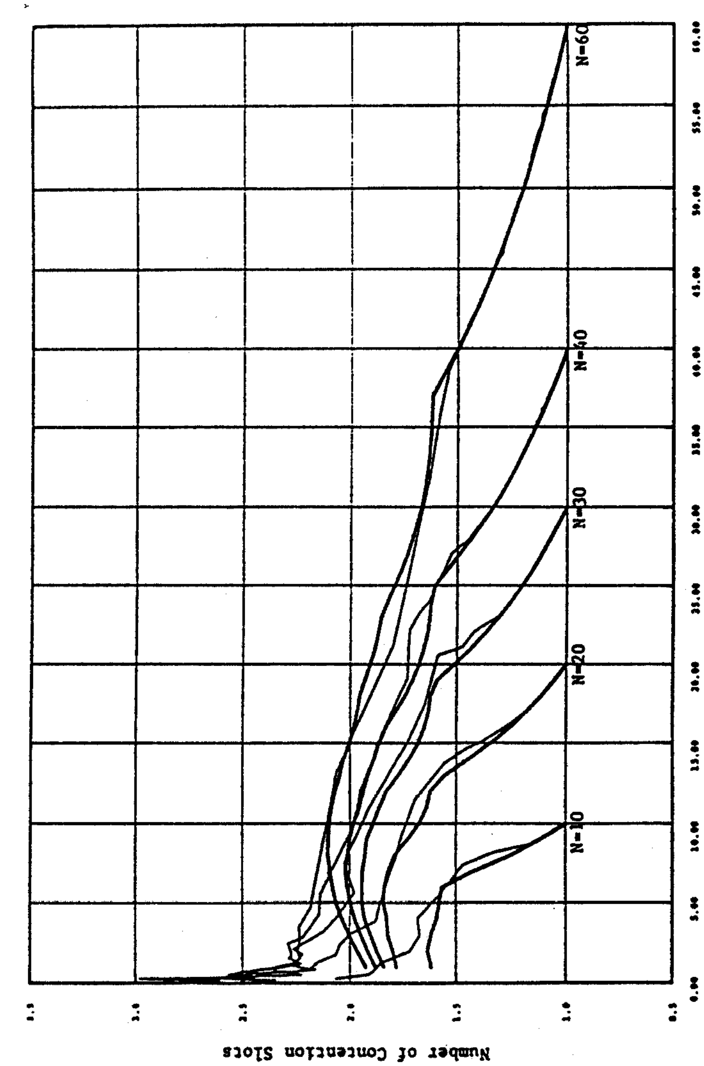


Fig. 2. Performance of adaptive-tree-polling and urn protocols degrades as the total number of stations in the network (N) increases.

A simulator driven by a random-number generator was implemented. Each station had a given probability p of becoming active in a time unit, but this probability was unknown to the window-control part of the simulator. The simulator used n, the observed channel load, to compute the distributions in Equation (3.9). The n is obtained by rounding up the moving average of the number of packets transmitted in each circumscription. The simulation results summarized in Figure 2 (thin lines) confirms our numerical evaluations very well except when channel loading is low. The mismatch at the low-load range is probably due to the large variance and roundup errors in estimating n. The consistency between the numerical evaluations and simulation results also verifies the correctness of the proposed method to estimate the distributions of contention parameters.

Numerical evaluations conducted indicate that the average number of contention iterations increases as N, the total number of stations in the network, increases (see Figure 2). This phenomenon confirms the postulate that higher certainty in generating contention parameters leads to better performance. In the extreme case when $N \to \infty$, the discrete protocols would behave like an infinite-population network with continuous contention parameters.

5. HEURISTIC WINDOW-CONTROL ALGORITHMS AND IMPLEMENTATIONS

The dynamic-programming window control discussed in Sections 2 and 4 provides a lower bound on the number of contention iterations. However, the computational complexity to compute the sequence of optimal windows is high; hence the method is impractical for real-time applications. As an example, the execution time to evaluate Equation (2.2) on a DEC VAX 11/780 computer is 1.3 seconds for $n = 20$, and increases to 828 seconds for $n = 100$. Efficient implementations are necessary to make the proposed optimization scheme practical.

In this section, heuristic methods for window control are proposed, and some implementation issues are discussed.

5.1. GREEDY ALGORITHMS

The optimization of window control using dynamic programming requires a high computational overhead, because it examines the entire sequence of possible future windows to determine a single window. To reduce this overhead, only one future window may be examined. An optimal greedy window-control scheme is one that finds a window to maximize the probability of success $g(w, a, b, n)$ in the next iteration. When contention parameters have *identical*

continuous distributions $F(x$ …
as

$$g(w, a, b, n) = K[F(w) \ldots$$

where $K = n/\{\Pr(A)[1 - F \ldots$
unimodal between a and b, …
the optimal value of w in E…
solve for w. This leads to t…

$$[1 - F(w)]^{n-1} - [1 - F(\ldots$$

Let $z = 1 - F(w)$. Equation …

$$z^{n-1} - \frac{(n-1)\ldots}{}$$

It can be shown that a r…
inequality $1 - F(b) < z_0 < 1$…
tion (5.3), and z_0 has to be …
upper boundary of the win…

The performance of th…
simulations is suboptimal a…
contentions (see Figure 3).

The computational overh…
dent of n and is less than o…
most cases. The algorithm…
improve the computational…
scheme using an approxim…
Equation (5.1) may be writ…

$$g(w, a, b, n) = K[F(w) - \ldots$$

whee $v = [1 - F(b)]/[1 - F$ …
has a maximum very close t…

continuous distributions $F(x)$, $g(w, a, b, n)$ can be expressed in a simpler form as

$$g(w, a, b, n) = K[F(w) - F(a)]\{[1-F(w)]^{n-1} - [1-F(b)]^{n-1}\}, \quad (5.1)$$

where $K = n/\{\Pr(A)[1-F(a)]^n\}$. It can be shown that Equation (5.1) is unimodal between a and b, so a maximum exists in the interval $(a, b]$. To find the optimal value of w in Equation (5.1), we set $\{\partial[g(w, a, b, n)]/\partial w\} = 0$ and solve for w. This leads to the following equation if $f(w) \neq 0$:

$$[1-F(w)]^{n-1} - [1-F(b)]^{n-1} = (n-1)[F(w) - F(a)][1-F(w)]^{n-2}. \quad (5.2)$$

Let $z = 1 - F(w)$. Equation (5.2) becomes

$$z^{n-1} - \frac{(n-1)[1-F(a)]z^{n-2}}{n} - \frac{[1-F(b)]^{n-1}}{n} = 0. \quad (5.3)$$

It can be shown that a real root of Equation (5.3) exists and satisfies the inequality $1 - F(b) < z_0 < 1 - F(a)$. There is no closed-form solution to Equation (5.3), and z_0 has to be solved for numerically. Once z_0 is obtained, w_0, the upper boundary of the window, can be computed directly from z_0 as

$$w_0 = F^{-1}(1 - z_0). \quad (5.4)$$

The performance of the optimal greedy window control as evaluated by simulations is suboptimal and approaches an average of 2.7 iterations to resolve contentions (see Figure 3).

The computational overhead to solve Equation (5.3) numerically is independent of n and is less than one second of CPU time on the DEC VAX 11/780 in most cases. The algorithm is still impractical for real-time applications. To improve the computational overhead, an approximate greedy window-control scheme using an approximate equation on success probability can be applied. Equation (5.1) may be written as

$$g(w, a, b, n) = K[F(w) - F(a)][F(b) - F(w)][1-F(w)]^{n-2}\sum_{i=0}^{n-2} v^i, \quad (5.5)$$

whee $v = [1 - F(b)]/[1 - F(w)]$. An approximation function $\hat{g}(w, a, b, n)$ that has a maximum very close to that of $g(w, a, b, n)$ can be found by replacing the

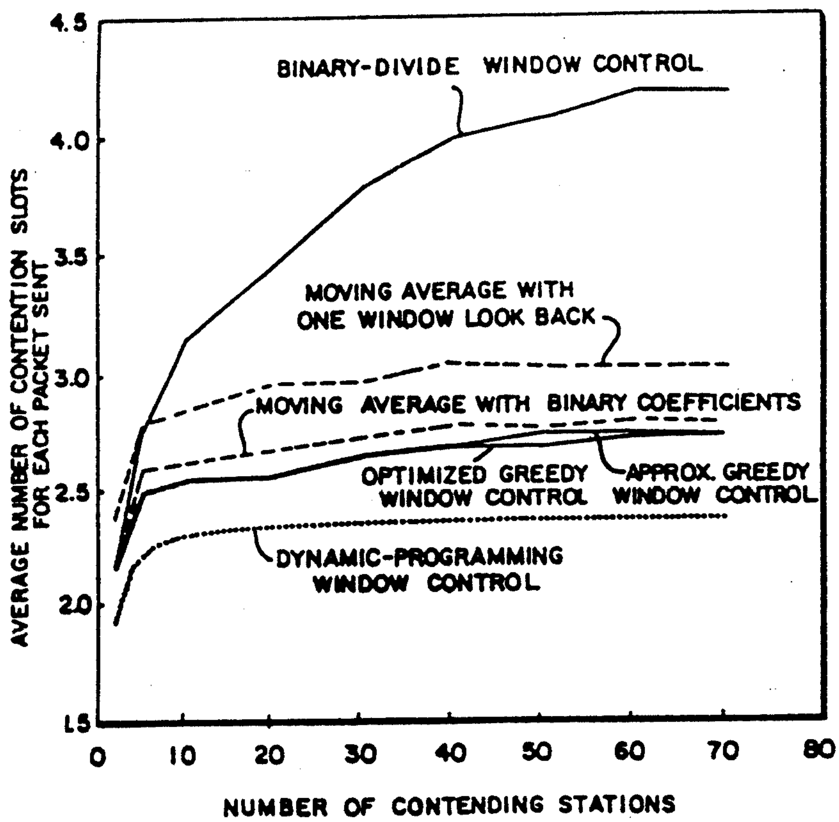


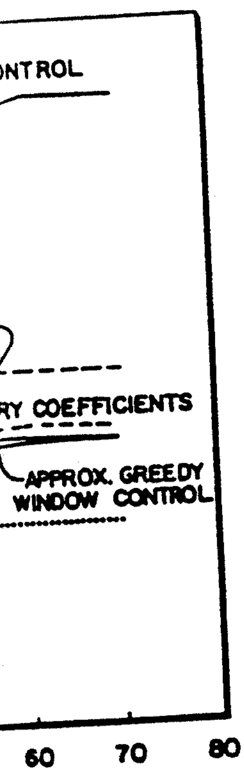
Fig. 3. Performance of infinite-population window protocols with different heuristic window-control and load-estimation methods. Solid lines indicate that the channel load is exactly known; dashed or dotted lines indicate that the channel load is estimated from previous experience. The distribution of contention parameters is uniform in (0,1].

sum $\Sigma_{i=0}^{n-2}v^i$ with $n-1$. That is,

$$\hat{g}(w,a,b,n) = K'[F(w)-F(a)][F(b)-F(w)][1-F(w)]^{n-2}, \quad (5.6)$$

where $K' = (n-1)K$. By solving $\partial\{\log_e \hat{g}(w,a,b,n)\}/\partial w = 0$, we obtain

$$\frac{f(w)}{F(w)-F(a)} + \frac{f(w)}{F(w)-F(b)} + \frac{(n-2)f(w)}{F(w)-1} = 0,$$



ls with different heuristic
te that the channel load is
nel load is estimated from
uniform in (0.1].

$)][1-F(w)]^{n-2}$, (5.6)

$/\partial w = 0$, we obtain

$\frac{2)f(w)}{v)-1} = 0,$

or equivalently,

$$[F(w)]^2 + C[F(w)] + D = 0, \tag{5.7}$$

where

$$C = \frac{-(n-1)[F(a)+F(b)]+2}{n},$$

$$D = \frac{F(a)+F(b)+(n-2)F(a)F(b)}{n}.$$

A solution to Equation (5.7) in the interval $(F(a), F(b)]$ is given by

$$F(w_a) = \frac{-C-\sqrt{C^2-4D}}{2}. \tag{5.8}$$

The approximate window w_a as computed from Equation (5.8) gives a performance that is nearly as good as that of the optimal greedy scheme (see Figure 3). The computational overhead to compute Equation (5.8) is also independent of n, and can be done in less than 100 μs on the DEC VAX 11/780.

The accuracy of the channel-load estimation using the equations (3.18) has been examined by comparing the performance of the optimal greedy window-control scheme with known loads and that with estimated loads. It turns out that the average number of iterations to resolve contentions using single-window lookback is 3.1, and the performance of using ARMA load estimation is very close to the case when the channel load is exactly known (see Figure 3).

When the distributions are nonidentical, the corresponding equation on success probability is very complex and cannot be optimized in real time.

5.2. *BINARY-DIVIDE ALGORITHM WITH LOAD ESTIMATION*

In a binary-divide window-control scheme, a transmission window is obtained by dividing a collided window into two equal halves. It is a robust algorithm even when nonidentical distributions are used to generate the contention parameters. It represents a class of CSMA protocols that include Kleinrock and Yemini's urn protocol, tree-polling algorithms [28, 15, 40], Gallagher's window protocol, and Ethernet's binary-exponential backoff algorithm. It has been shown that a pure binary-divide algorithm has an average time complexity bounded by $O(\log n)$ iterations, where n is the number of contending stations [46]. Simulation results in Figure 3 also confirm this result.

In many protocols, an initial window is generated with an estimated channel load, and binary-divide is used in subsequent iterations. The initial window should be properly chosen so that the number of contending stations in the window is relatively small even if collision is not resolved. As shown in Figure 3, a binary-divide scheme performs satisfactorily when the number of contending stations is small. In particular, the binary-divide scheme is optimal when there are two contention parameters in a collided window.

5.3. *BINARY DECISION TREE AND ITS IMPLEMENTATION*

Assuming identical and continuous distributions to generate the contention parameters, the sequence of windows evaluated by dynamic programming can be precomputed and stored in a lookup table. Given a channel load n and distribution $F(\cdot)$, the sequence of optimal windows derived from Equation (2.2) constitute a binary decision tree [Figure 4(a)]. The root of a subtree represents a window. The optimal window for the next iteration will reside in the left subtree if collision is detected in the current slot. It will be in the right subtree if no transmission is detected. A set of binary trees, each of which corresponds to a channel load, can be constructed and stored as a lookup table in each station. The data structure to implement the binary decision tree is shown in Figure 4b. The optimal window in each contention slot can therefore be retrieved efficiently in real time.

One problem with the lookup-table method lies in the large memory space required. Since the average number of contention slots is small, some subtrees that are seldomly visited can be pruned to reduce the memory space without significant degradation in performance. Window in the pruned subtrees can be obtained by interpolation techniques or simply by binary-divide schemes. Likewise, for channel loads for which no decision trees are stored, interpolation has to be used to obtain window boundaries.

Existing Ethernet interfaces have to be modified for implementing the look-up table method. A microcontroller, Intel MCS 8396, is placed between the Ethernet-protocol chip, Intel 82586, and the collision-detection chip, Intel 82501. Sixteen-bit random numbers are used for the contention parameters and the entries of the decision tree.

Two alternatives for storing the decision tree have been investigated. First, a decision tree of four levels as evaluated by dynamic programming is used, and the microcontroller switches to binary-divide window control when more than four contention slots are needed. The channel load is assumed to vary from one to 100 stations. Hence, the total space required for storing the lookup table is 3 Kbytes, which can fit in the 8-Kbyte read-only memory of the MCS 8396. The performance of the truncated decision-tree method is less than 3.0 contention

with an estimated channel
tions. The initial window
contending stations in the
olved. As shown in Figure
n the number of contend-
e scheme is optimal when
ndow.

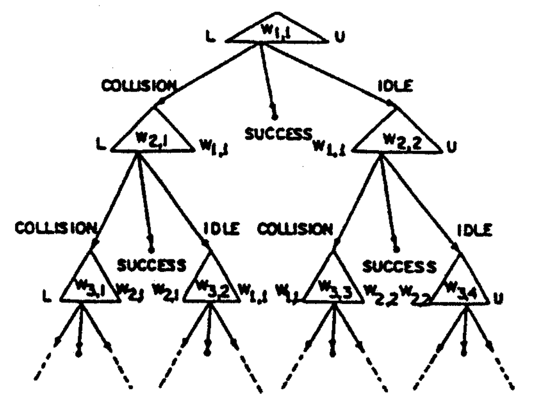
TION

to generate the contention
dynamic programming can
en a channel load n and
derived from Equation (2.2)
ot of a subtree represents a
on will reside in the left
ill be in the right subtree if
ch of which corresponds to
okup table in each station.
tree is shown in Figure 4b.
therefore be retrieved effi-

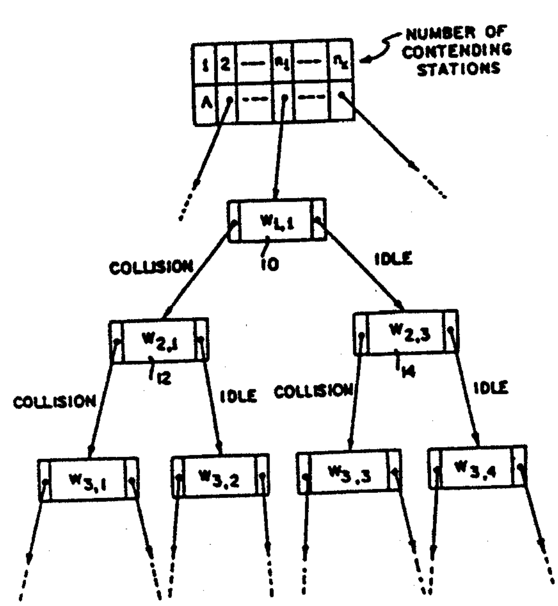
in the large memory space
lots is small, some subtrees
the memory space without
the pruned subtrees can be
by binary-divide schemes.
ees are stored, interpolation

fied for implementing the
CS 8396, is placed between
llision-detection chip, Intel
contention parameters and

e been investigated. First, a
c programming is used, and
ow control when more than
is assumed to vary from one
storing the lookup table is 3
mory of the MCS 8396. The
l is less than 3.0 contention



(a) Binary decision tree.



(b) Corresponding data structure.

Fig. 4. Lookup-table implementation of dynamic-programming window control: (a) binary decision tree, (b) corresponding data structure.

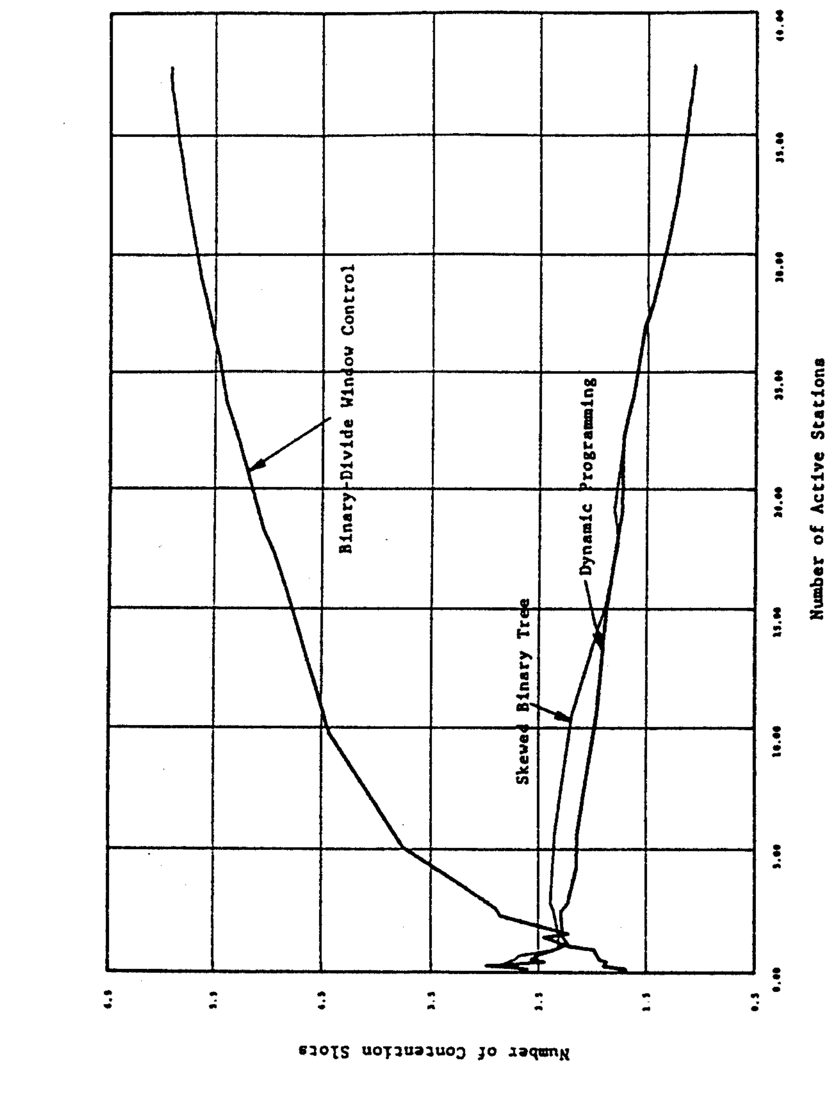


Fig. 5. A comparison between full dynamic-programming tree and skewed binary tree to implement adaptive-tree-polling protocols (total number of stations in the network is 40).

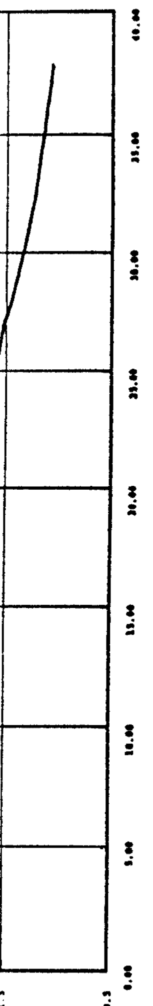


Fig. 5. A comparison between full dynamic-programming tree and skewed binary tree to implement adaptive-tree-polling protocols (total number of stations in the network is 40).

slots (assuming that the channel load is exactly known). Although the balanced binary tree simplifies the data structure, the performance is worse than optimal. In the second alternative, a skewed binary tree is used. The reasoning behind the skewed tree is that when a collision occurs, the left subtree is traversed and the size of the interval containing the minimum contention parameter is small. In this case, a binary-divide control works well. On the other hand, when no transmission is detected, the right subtree is traversed and the size of the interval containing the minimum is not reduced significantly. In this case, the binary-divide control does not work well. Experimental results indicate that less than 2.5 slots are required to resolve a contention when a skewed binary tree with a height equal to n and a height of one for the left subtree of every nonterminal node (n is exactly known) is used. This means that $2n$ words are required for every dynamic-programming tree. The total memory space required for n ranging from 1 to 60 is 7.3 Kbytes.

The skewed-binary-tree approach can also be applied to adaptive-tree-polling protocols. The simulation results summarized in Figure 5 show that a protocol using skewed binary tree can perform nearly as well as the one that uses full dynamic programming tree. The evaluations of a pure binary-divide control scheme for adaptive-tree-polling protocols are also shown in the figure for comparison.

6. CONCLUDING REMARKS

In this paper we have described a window-search procedure to find the minimum among a set of random numbers. This procedure provides a unified framework for resolving contentions in CSMA and CSMA/CD networks. It is shown that many existing protocols can be mapped to this procedure. Thus, the optimization scheme developed for this procedure can be applied to every protocol in this class for both finite-population and infinite-population networks.

There is clearly a tradeoff between the performance of a contention-resolution protocol and the amount of dynamic information to be collected and used. We started with the objective of designing a protocol that has a load-independent performance and that does not require additional dynamic information to be collected besides the information already available on the bus. We formulated the problem in a dynamic programming equation and minimized the contention overhead to transmit each packet. In the simplest case, the dynamic information needed is the distribution of the contention parameters and the current channel load. Assuming that this information is known exactly, the proposed protocol has load-independent performance. We have used two simple heuristics to estimate the channel load while assuming that the distributions of contention parameters are known exactly, and have found that the performance is still load-independent. The dynamic information needed also depends on the particular protocol considered and whether the distributions to generate the contention parameters are identical or not. In any case, we have found that even when this information has to be estimated from long-term averages, the system still performs satisfactorily. The optimal derivation of the minimum amount of dynamic information to achieve the maximum throughput remains to be studied.

Our results can be summarized as follows. First, the effect of truncation on the evaluation of the dynamic programming formulation with continuous distributions is insignificant if the truncation interval is small enough. Second, numerical evaluations suggest an asymptotic bound of 2.4 average contention iterations for resolving contentions in infinite-population networks. Lastly, for finite-population networks with discrete contention parameters, nearly perfect scheduling can be achieved under heavy traffic conditions.

The proposed optimization method has the following advantages. First, it is optimized with respect to instantaneous channel load, and hence can be more adaptive than those that optimize the performance with respect to the average load. Second, it is independent of the type of distributions, and hence can be applied to both discrete and continuous cases. Third, it is able to cope with traffic patterns other than Poisson arrivals. Lastly, it allows the instantaneous channel load to be estimated from the result of previous contentions.

REFERENCES

1. K. Arrow, L. Pesotchinsky, and M. Sobel, On partitioning a sample with binary-type questions in lieu of collecting observations, *J. Amer. Statist. Assoc.* 76(374):402–409 (June 1981).
2. K. M. Baumgartner and B. W. Wah, The effects of load balancing on response time for local computer systems with a multiaccess network, in *Proceedings of the IEEE International Conference on Communications*, IEEE, June 1985, pp. 10.1.1–10.1.5.
3. T. Berger, N. Mehrauari, D. Towsley, and J. K. Wolf, Random multiple access and group testing, *IEEE Trans.* COM-34(7):769–779 (July 1984).
4. J. Capetanakis, The Multiple Access Broadcast Channel: Protocol and Capacity Considerations, Ph.D. Thesis, Massachusetts Inst. of Technology, 1977.
5. J. Capetanakis, Tree algorithm for packet broadcast channels, *IEEE Trans. Inform. Theory* IT-25(5):505–515 (Sept. 1979).
6. J. Capetanakis, Generalized TDMA: The multi-accessing tree protocol, *IEEE Trans. Comm.* COM-27:1479–1484 (Oct. 1979).
7. I. Chlamtac and W. R. Franta, "Massage-based priority access to local networks, *Comput. Comm.* 3(2):77–84 (Apr. 1980).
8. E. J. Coyle and B. Liu, Throughput, Delay, and Stability of Asynchronous Random Multiple Access Protocols, Technical Report, EECS Dept., Princeton Univ., 1982.
9. E. J. Coyle and B. T. Liu, Finite population CSMA/CD networks, *IEEE Trans. Comm.* COM-31 (11):1247–1251 (Nov. 1983).

ave found that the perfor-
ation needed also depends
ie distributions to generate
y case, we have found that
m long-term averages, the
erivation of the minimum
imum throughput remains

the effect of truncation on
nulation with continuous
l is small enough. Second,
of 2.4 average contention
ation networks. Lastly, for
parameters, nearly perfect
itions.
ing advantages. First, it is
d, and hence can be more
vith respect to the average
outions, and hence can be
d, it is able to cope with
t allows the instantaneous
ous contentions.

ng a sample with binary-type
tatist. *Assoc.* 76(374):402–409

palancing on response time for
oceedings of the IEEE Interna-
. 10.1.1–10.1.5.
dom multiple access and group

otocol and Capacity Consider-
977.
annels. *IEEE Trans. Inform.*

tree protocol, *IEEE Trans.*

cess to local networks, *Comput.*

ty of Asynchronous Random
Princeton Univ., 1982.
networks, *IEEE Trans. Comm.*

10. R. Cruz and B. Hajek, A new upper bound to the throughput of a multi-access broadcast channel, *IEEE Trans. Inform. Theory* IT-28, No. 3 (May 1982).
11. G. Fayolle et al., Stability and optimal control of the packet switching broadcast channel, *J. Assoc. Comput. Mach.* 24(3):375–386 (July 1977).
12. R. G. Gallagher, Conflict resolution in random access broadcast networks, in *Proceedings of the AFOSR Workshop on Communication Theory and Applications*, 17–20 Sept 1978, pp. 74–76.
13. R. G. Gallagher, A perspective on multiaccess channels, *IEEE Trans. Inform. Theory* IT-31(2):124–142 (Mar. 1985).
14. Y. I. Gold and W. R. Franta, An efficient collision-free protocol for prioritized access-control of cable radio channels, in *Computer Networks*, Vol. 7, North-Holland, 1983, pp. 83–98.
15. E. Gudjohnsen, D. Towsley, and J. K. Wolf, On adaptive polling techniques for computer communication networks, in *Proceedings of the IEEE International Conference on Communications*, Vol. 1, 1980, pp. 13.3.1–13.3.5.
16. B. E. Hajek, Information of partitions with applications to random access communications, *IEEE Trans. Inform. Theory* IT-28(5):691–701 (Sept. 1982).
17. J. H. Hayes, An adaptive technique for local distribution, *IEEE Trans. Comm.* COM-26, No. 8 (Aug. 1978).
18. M. G. Hluchyj, Multiple Access Communication: The Finite User Population Problem, Tech. Report LIDS-TH-1162, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 1981.
19. Y. C. Jeng, On the stability of slotted aloha systems, *IEEE Trans. Comm.* COM-28, No. 11 (Nov. 1980).
20. J. Y. Juang and B. W. Wah, Unified window protocol for local multiaccess networks, in *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, Apr. 1984, pp. 97–104.
21. J. Y. Juang, Resource Allocation in Computer Networks, Ph.D. Thesis, Purdue Univ., West Lafayette, Ind., Aug. 1985.
22. L. Kleinrock and F. A. Tobagi, Packet switching in radio channels: Part I—Carrier sense multiple access modes and their throughput-delay characteristics, *IEEE Trans. Comm.* COM-23(12):1400–1416 (Dec. 1975).
23. L. Kleinrock and Y. Yemini, An optimal adaptive scheme for multiple access broadcast communication, in *Proceedings of the IEEE International Conference on Communications*, IEEE, 1978, pp. 7.2.1–7.2.5.
24. J. F. Kurose and M. Schwartz, A family of window protocols for time constrained applications in CSMA networks, in *Proceedings of the 2nd Joint Conference of Computer and Communication Societies*, IEEE, 1983, pp. 405–413.
25. J. F. Kurose, M. Schwartz, and Y. Yemini, Multiple-access protocols and time-constrained communication, *ACM Comput. Surveys* 16(1):43–70 (Mar. 1984).
26. R. M. Metcalfe and D. R. Boggs, Ethernet: Distributed packet switching for local computer networks, *Comm. ACM* 19(7):395–404 (July 1976).
27. K. Mittal and A. Venetsanopoulos, On the dynamic control of the urn scheme for multiple access broadcast communication systems, *IEEE Trans. Comm.* COM-29:962–970 (July 1981).
28. A. K. Mok and S. W. Ward, Distributed broadcast channel access, in *Computer Networks*, Vol. 3, North-Holland, 1979, pp. 327–335.
29. M. L. Molle, On the capacity of finite population multiple access protocol, *IEEE Trans. Inform. Theory* IT-28(3):396–401 (May 1982).
30. J. Mosely, An Efficient Contention Resolution Algorithm for Multiple Access Channels, M.S. Thesis, Dept. of EECS, Mass. Inst. of Tech., May 1982.