UNIFIED WINDOW PROTOCOLS FOR CONTENTION RESOLUTION IN LOCAL MULTI-ACCESS NETWORKS

Jie-Yong Juang and Benjamin W. Wah
School of Electrical Engineering
Purdue University
W. Lafayette, IN 47907

ABSTRACT

In this paper we have unified a class of adaptive CSMA protocols that include the Adaptive-Tree-Walk Protocol, the Urn Protocol, the Priority-Access Protocol, the Arrival-Time-Window Protocol and the Virtual-Window Protocol. Several common characteristics are identified in these protocols: (a) each station in the network generates a contention parameter that is governed by a possibly station-dependent distribution function; (b) a global window is maintained in all the stations; and (c) a distributed window-control scheme is used to find the extremum of the contention parameters. A unified window protocol is proposed to encompass these different protocols. Optimal control of the window sizes can be obtained by dynamic programming which minimizes the expected total number of contention slots. Heuristic schemes that are computationally efficient and accurate are also proposed.

KEYWORDS AND PHRASES: Adaptive-Tree-Walk Protocol, Arrival-Time-Window Protocol, CSMA/CD network, contention, dynamic programming, priority, Urn Protocol, Virtual-Window Protocol.

1. INTRODUCTION

A Carrier-Sense-Multiple-Access/Carrier-Detect (CSMA/CD) network is characterized by a single shared channel with a distributed contention-resolution protocol. The unit of message is a packet and can be sent in a packet time. A station is active if it is attempting to access the channel; otherwise, it is idle. Before a stations can send a packet, it detects whether a carrier is present (the channel is being used). If the channel is busy, the station waits until it becomes free; otherwise, the station starts the transmission. When two stations try to transmit simultaneously, a *collision* is said to occur. The contending stations have to wait and transmit again in the future.

CSMA/CD networks have been studied for over a decade, and many protocols were proposed. These protocols can broadly be classified into nonadaptive and adaptive according to their transmission policies. A nonadaptive CSMA protocol makes its transmission decision regardless of the channel load. Nonpersistent and p-persistent protocols are examples of this class [KLE75a]. On the other hand, the knowledge of channel load is used to schedule transmission in an adaptive CSMA protocol. This knowledge may be estimated explicitly by a dedicated channel monitor as proposed by Kleinrock and Yemini [KLE78], or may be acquired implicitly by the contention experiences of an individual station such as the Binary Exponential Backoff scheme of Ethernet [MET76]. It has been shown that a CSMA network with nonadaptive protocols is inherently instable [FAY77]. The effective throughput of the channel will tend to zero due to the possible endless collisions. In order to maintain a stable channel, an adaptive protocol is necessary.

Many adaptive CSMA protocols were proposed, notably are the Adaptive-Tree-Walk Protocol [CAP79], the Urn Protocol [KLE78, MIT81], the Arrival-Time-Window Protocol [TOW82, KUR83] and the Virtual-Window Protocol [WAH83]. Other nonadaptive protocols for priority resolution have been studied [TOB82, NI83], but the corresponding adaptive protocols have not been found. In this paper we study these protocols and explore their common characteristics. It is shown that these protocols belong to a general class of window-control protocols in which station i generates a contention parameter that is governed by a station-dependent distribution. A unified model is proposed which defines a basic window-search procedure for resolving contentions and finding the extremum of the contention parameters. The objective function to be optimized in different protocols is translated into a common objective function that is expressed in terms of the distribution functions and the channel load. A single optimization method can, therefore, be applied to all these different protocols.

The paper is organized in the logical sequence of model development. In Section 2 the window-search procedure is presented, and the family of protocols that can be solved by this procedure are identified. In particular, the underlying distributions of the contention parameters for all the stations are derived. The dynamic-programming formulation is shown in Section 3. It is found that contention can be resolved in an expected time of about two contention slots regardless of the channel load. It should be pointed out that this result is based on knowledge of the distribution functions and the channel load. The performance is improved as compared to previous protocols that utilize the channel load alone, and can be said to be optimal as far as the order of magnitude is concerned. However, dynamic-programming methods are time-consuming, and more efficient methods are studied in Section 4. The problem of estimating channel load and global distributions are also discussed.

2. UNIFIED WINDOW PROTOCOL

2.1. Contention Resolution Model

In a multiaccess bus, simultaneous requests for channel access can be generated by independent stations. In order to resolve contention and to allocate exactly one

This research was partially supported by National Science Foundation Grant ECS80-16580 and by CIDMAC, a research unit of Purdue University, sponsored by Purdue, Cincinnati Milicron Corporation, Control Data Corporation, Cummins Engine Company, Ransburg Corporation and TRW.

Third Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, CA, 1984.

station the access of the channel, a contention-resolution protocol is necessary. An easy way is to ask every contender to generate a parameter, and the station with the minimum (or maximum) parameter wins the contention. From this point of view, the contention-resolution problem can be reduced to the problem of finding the extremum among a set of contention parameters.

A distributed window-search scheme is proposed here to solve the above problem. Suppose the set of contention parameters are $\{x_1, ..., x_n\}$ in the interval between L and U, and y_i is the i-th smallest of the x_j's. Given that the minimum value is sought, an initial interval is chosen with the lower bound at L and the upper bound between L and U. There can be zero, one or many y_i's lying in this interval. If there is exactly one number falling within the interval, this number can be verified as the minimum, y_1. Otherwise, the interval has to be updated: it is moved if it contains no y_i; or it is shrunk to a smaller size if it contains more than one y_i's. The process is repeated until the minimum is uniquely isolated in the interval. The procedure is outlined in Figure 1.

```
procedure search (n̂, y1, y2, ..., yn̂, window);
[
/* n̂ - number of elements,
   y1, ...,yn̂ - set of random numbers of distribution F(y),
   window - function to calculate window size w,
   lb_window - lower bound of window to be searched,
   ub_window - upper bound of window to be searched.
*/

lb_window = L;
ub_window = U;
w = window( lb_window, ub_window, n̂);

while (TRUE) do [
     if (y1 ≥ w and y2 ≥ w) then
          lb_window = w;    /* update lb */
     else
          if (y1 < w and y2 < w) then
               ub_window = w;    /* update ub */
          else [
               /* successful isolation of minimum */
               return(lb_window, ub_window);
               ]
     w = window( lb_window, ub_window, n̂);
     ]
]
```

Figure 1. Window based minimum-search procedure.

Implementation of the distributed window-search scheme on a CSMA/CD network requires a method of detecting the number of parameters covered by an interval. This can be provided by the collision-detection capability. A global window is kept by all the stations, and stations with parameters inside the window are allowed to transmit. The state of the channel due to the transmission can be either idle, successful or collision which corresponds respectively to zero, one or many minima lying in the window. Therefore, it is very efficient to implement the window-search procedure on CSMA/CD networks. The resulting protocol called *window protocol* should consist of the following components:

- *Contention parameters*: Each contending station generates a contention parameter that is governed by an application-dependent distribution function.
- *Global window*: A global window is maintained at each station, and it serves as a reference for the decision of transmission. An initial window must be selected before contention begins.
- *Transmission rule*: A station is allowed to transmit if its contention parameter is inside the window.
- *Distributed window control*: Window is updated in a distributed fashion according to outcome of transmission. The windows maintained by all the stations must be identical in each step of the contention process. This can be achieved if all the stations receive identical information and apply the same rule in updating the window.

2.2 The Family Of Window Protocols

In this section several adaptive CSMA/CD protocols are shown to be members of the family of window protocols. The distributions of contention parameters that characterize these protocols will be discussed here. The distributions of the first three protocols are discrete, whereas those of the last two are continuous.

(a) *Adaptive-Tree-Walk Protocol*

This protocol is basically a preorder binary-tree traversal algorithm [CAP79]. A binary tree is organized in such a way that each leaf is associated with a station in the network. The search begins at the root, and all ready stations can transmit in the first contention slot. If a collision occurs, the search continues recursively with the left and right subtrees in the following contention slots. The search stops when a single ready station is contained in a subtree. After the packet is transmitted, the next contention slot is reserved for the next subtree in the preorder search.

This protocol can be viewed as a moving window protocol in which the size of the window is equal to the number of leaves in the subtree. The objective is to isolate a station with the minimum distance from the origin of the window. Without loss of generality, suppose the origin is currently at station 1. A ready station i generates a contention parameter which is the distance from 1, while an idle station generates a very large number which is never considered by the protocol, say N+1. The probability density function from which a station generates its contention parameter is:

$$f_i(k) = \begin{cases} \Pr\{\text{i-th station is active}\} & k = i \\ \Pr\{\text{i-th station is idle}\} & k = N+1 \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

Suppose there is only one buffer at each station. Let p be the probability of a new packet arriving in a packet-transmission time (assuming that contention time is much smaller than packet time) and t_i be the number of packet times elapsed since the i-th station was allowed to transmit, then

$$\Pr\{\text{i-th station is active} \mid p, t_i\} = 1 - (1-p)^{t_i} \quad (2.2)$$

p is a function of N, the total number of stations in the network, and n, the number of stations that have transmitted when the window has circumscribed over all the stations once. The next station allowed to transmit must have been idle for n packet times, and the probability that a packet is ready is $(1-(1-p)^n)$. For the N stations in the network, the expected number that transmit is:

$$N\,(1 - (1-p)^n) = n \quad (2.3)$$

Solving Eq. 2.3 yields:

$$p = 1 - \left(1 - \frac{n}{N}\right)^{\frac{1}{n}} \qquad (2.4)$$

t_i cannot be found exactly, but can be estimated from n. In order to compute t_i, it is necessary to know the minimum and 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. The number of active stations between station i and the origin has an incomplete binomial distribution with probability (N−i)/N.

$$\Pr(t_i=k) = \frac{\binom{n}{k}\left(\frac{N-i}{N}\right)^k\left(1-\frac{N-i}{N}\right)^{n-k}}{\sum_{j=\max(0,n-i)}^{\min(n,N-i)}\binom{n}{j}\left(\frac{N-i}{N}\right)^j\left(1-\frac{N-i}{N}\right)^{n-j}} \qquad (2.5)$$

The equations derived here are approximations because they assume that each station has constant elapsed time before it is allowed to transmit.

Substituting Eq's 2.2, 2.4 and 2.5 into Eq. 2.1, the distribution function for station i, $1 \le i \le N$, to generate its contention parameter is:

$$F_i(k) = \begin{cases} 0 & k < i \\ 1 - \sum_{j=\max(0,n-i)}^{\min(n,N-i)}\left(1-\frac{N-i}{N}\right)^{j/n}\Pr(t_i=j) & i \le k \le N \\ 1 & k > N \end{cases} \qquad (2.6)$$

A window is defined in the interval [1,N] such that stations with contention parameters inside the window can contend. The adaptive-tree-walk protocol is a heuristic way of controlling the window size by halving or doubling it after each contention.

(b) *Urn Protocol*

The Urn Protocol of Kleinrock and Yemini [KLE78] is similar to the Adaptive-Tree-Walk Protocol in that each ready station generates a contention parameter which is the distance from the origin of the window. The distribution function given in Eq. 2.6, thus, applies. The difference lies in the window control. In the Urn Protocol, the initial window size, w, is chosen by optimizing the urn model. During each contention slot, those stations inside the window are given permission to transmit. The window is advanced w positions if there is successful or no transmission. In case of collision, the window is shrunk to half of its original size. This window-control scheme is applicable when the set of ready stations are uniformly distributed. 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. A better control scheme must be developed.

(c) *Priority-CSMA Protocols*

Several nonadaptive CSMA protocols for handling priority messages have been suggested in recent years. Tobagi proposed that priorities be resolved linearly [TOB82,SHA83]. Each station is assigned the highest priority of the local messages. During the resolution of priorities, a slot is reserved for each priority class. A ready station contends during the slot reserved for the local priority. The process stops when the highest priority level is determined. This scheme is good when high priority messages are predominantly sent. Subsequently, Ni developed a binary-divide scheme which resolves the highest priority level in $O(\log_2 P)$ steps where P is the maximum number of priority levels [NI83]. This assumes that the highest priority level is equally likely to be any one of the P priority levels.

A closer look at the problem reveals that the highest priority class of the stations is sought. This is the problem of finding the maximum priority from a set of n contending stations (priority classes ranging from 1 to P). It can be solved by a window protocol such that the highest priority class at each station is used as the contention parameter [WAH83].

In pursuing distributions of priority classes, it is assumed that the message arrival rate of priority class i at station j is $\lambda_{i,j}$ ($\lambda_i = \sum_{j=1}^{N}\lambda_{i,j}$), and the corresponding service rate is $\mu_{i,j}$. Suppose the channel can service a message of class i at rate μ_i. Since a lower priority message can only be serviced during an idle period of servicing higher priority messages, the effective service rate of class i messages in the channel is:

$$\mu_i^e = \mu_i \prod_{k=i+1}^{P}(1-\rho_k) \qquad i=1, 2,..., P \qquad (2.7)$$

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

$$\rho_i = \frac{\lambda_i}{\mu_i^e} \qquad i=1, 2, ..., P \qquad (2.8)$$

From the results of queueing theory [KLE75b], the effective service rate of class i messages at node j is:

$$\mu_{i,j}^e = \mu_i^e \frac{\lambda_{i,j}}{\lambda_i} \qquad i=1, ..., P; j=1, ..., N \qquad (2.9)$$

The traffic intensity of class i messages at node j is:

$$\rho_{i,j} = \frac{\lambda_{i,j}}{\mu_{i,j}^e} \qquad i=1, ..., P; j=1, ..., N \qquad (2.10)$$

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

$$f_j(p) = \rho_{p,j}\prod_{i=p+1}^{P}(1-\rho_{i,j}) \qquad j=1, ..., N; p=1, ..., P \qquad (2.11)$$

The distribution of the contention parameter at node j is:

$$F_j(p) = \sum_{k=1}^{p} f_j(k) \qquad j=1, ..., N; p=1, ..., P \qquad (2.12)$$

(d) *Arrival-Time-Window Protocol*

Towsley proposed a window protocol on the time axis in which all stations have a common window of length u in the past [TOW82]. Stations with packets arriving during the window are allowed to contend. If there is no or a successful transmission, the window is advanced u units of time; 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. However, they do not reflect instantaneous load conditions.

Suppose the window begins at time O, the current time is T, and there are n contending stations. Since Towsley's protocol searches for the earliest packet arrival time in this window, each station can use the first packet-arrival time in the interval (O,T) as the contention parameter. Assuming that the arrival process at each station to be Poisson, the distribution of the first

arrival time conditioned on the current time T and the origin of window O is:

$$F_i(t \mid O<t\le T) = \begin{cases} 0 & t\le O \\ \dfrac{1-e^{-\lambda_i(t-O)}}{1-e^{-\lambda_i(T-O)}} & O<t\le T \quad i=1,\ldots,N \\ 1 & t>T \end{cases} \tag{2.13}$$

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

(e) *Virtual-Window Protocol*

A new Virtual-Window Protocol has been proposed to resolve contentions of messages [WAH83]. Each of the n active stations generates a random number from the uniform distribution U(0,1) as its contention parameter. That is,

$$F_i(y) = \begin{cases} 0 & y\le 0 \\ y & 0<y\le 1 \\ 1 & y>1 \end{cases} \quad i=1,\ldots,N \tag{2.14}$$

A contention parameter is only a dummy argument in this protocol, and no physical meaning is attributed.

3. OPTIMIZATION OF WINDOW CONTROL

A successful transmission on a multiaccess channel is always preceded by a contention period which resolves the channel-access right. For most applications, contention periods are independent, therefore, minimizing the length of individual contention period tends to optimize the overall performance of the channel. The length of a contention period for the window protocols described in this paper is the number of contention slots expended before the extremum is exclusively identified. In optimizing window control, a sequence of windows for each step of contention have to be found so that the expected number of contention slots is minimized. In this section an optimal algorithm based on dynamic programming is presented.

3.1 Optimal Window Control By Dynamic Programming

The minimization of the expected number of contention slots depends not only on the probability of success in the current slot, but also on the number of future contention slots in case that transmission is unsuccessful in the current slot. The formulation requires the following definitions:

- n(a,b) — the minimum expected number of contention slots to resolve contention given that all contention parameters are in the interval (a,U] and collision occurs in the current window (a,b];
- g(w,a,b) — probability of *success* in the next contention slot if a window of (a,w], a<w<b, is used;
- ℓ(w,a,b) — probability of *collision* in the next contention slot if a window of (a,w], a<w<b, is used;
- r(w,a,b) — probability of *no transmission* in the next contention slot if a window of (a,w], a<w<b, is used.

Note that:

$$\ell(w,a,b) + g(w,a,b) + r(w,a,b) = 1 \tag{3.1}$$

The problem of optimizing window control can be formulated recursively as follows:

$$n(a,b) = \min_{a<w<b} \left\{ \frac{1 + 0\cdot g(w,a,b)}{+\, n(a,w)\cdot\ell(w,a,b) + n(w,b)\cdot r(w,a,b)} \right\} \tag{3.2}$$

The probabilities g(w,a,b), ℓ(w,a,b) and r(w,a,b) can be derived from the distributions of the contention parameters. When transmission is unsuccessful, it is always possible to identify an interval (a,b] such that at least two of the x_i's lie in (a,b] and no x_i is smaller than a. This condition is designated as event A.

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

Suppose the window is reduced to (a,w], a<w<b, in the next slot, three mutually exclusive events corresponding to the three possible outcomes in the next slot can be identified:

B = {exactly one x_i's is in (a,w]};
C = {no x_i is in (a,w]};
D = {more than one x_i's are in (a,w]}.

From these events, the probabilities can be derived as:

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

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

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

Let $F_i(x)$ be the distribution function for generating x_i, $1\le i\le N$, and M be the number of stations that are contending (M=N for the Tree-walk or Urn Protocols, M=n for other protocols), then event A has probability:

$$\Pr(A) = \prod_{i=1}^{M}[1-F_i(a)] - \sum_{i=1}^{M}\left\{[F_i(b)-F_i(a)]\prod_{\substack{j=1\\ j\neq i}}^{M}[1-F_j(b)]\right\} - \prod_{i=1}^{M}[1-F_i(b)] \tag{3.3}$$

The first and last terms indicate 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 interval (a,b]. Similarly,

$$g(w,a,b) = \frac{1}{\Pr(A)}\sum_{i=1}^{M}[F_i(w)-F_i(a)] * \left\{\prod_{\substack{j=1\\ j\neq i}}^{M}[1-F_j(w)] - \prod_{\substack{j=1\\ j\neq i}}^{M}[1-F_j(b)]\right\} \tag{3.4}$$

$$r(w,a,b) = \frac{1}{\Pr(A)}\left\{\prod_{i=1}^{M}[1-F_i(w)] - \sum_{i=1}^{M}\left[[F_i(b)-F_i(w)]\prod_{\substack{j=1\\ j\neq i}}^{M}[1-F_j(b)]\right] - \prod_{i=1}^{M}[1-F_i(b)]\right\} \tag{3.5}$$

It follows that given the distributions of contention parameters, an optimal window can be derived in each step of the contention process by finding a value of w which minimizes n(a,b) in Eq. 3.2.

3.2 Numerical Evaluations

The values of n(a,b) are computed with respect to the various CSMA/CD protocols discussed in Section 2.2.

(a) *Virtual-Window Protocol*

Since the distributions of the contention parameters are independent and uniformly distributed over (0,1), Eq's 3.4 and 3.5 can be reduced to simpler forms:

$$g(w,a,b) = \frac{(w-a)\{(1-w)^{n-1}-(1-b)^{n-1}\}}{(1-a)^n-(1-b)^n-n(b-a)(1-b)^{n-1}} \quad (3.6)$$

$$r(w,a,b) = \frac{(1-w)^n-(1-b)^n-n(b-w)(1-b)^{n-1}}{(1-a)^n-(1-b)^n-n(b-a)(1-b)^{n-1}} \quad (3.7)$$

Although these equations simplify evaluation of Eq. 3.2, the dynamic-programming formulation is continuous and requires infinite levels of recursion. Some boundary conditions must be known in order to stop the evaluation after some levels. Suppose the x_i's are never too close together so that contention can always be resolved in one step whenever the window size is smaller than δ.

$$n(a,b) = 1 \qquad \text{for all } b-a < \delta \quad (3.8)$$

where δ is a small positive number. It was set to one tenth of the number of contending stations in our evaluations. The results of evaluation are plotted in Figure 4 which shows that the average number of contention slots is bounded by 2.3 and is independent of channel load. This is a desirable property that is not achievable by any existing protocol.

(b) *Arrival-time Window Protocol*:

The contention parameter x_i of the Arrival-time Window Protocol is generated by an incomplete exponential distribution. A random variable generated by such a distribution, $F_i(\cdot)$, can be transformed into another random variable that is uniformly distributed over (0,1) by replacing x_i with [PAP65]:

$$x_i' = F_i(x_i) \quad (3.9)$$

Since this transformation is a one-to-one mapping and if the distributions are identical, the optimization performed on the transformed contention parameters can be shown to be equivalent to the original optimization. On the other hand, if the distributions are non-identical, some properties are lost after the transformation. For example, the original order of packet arrivals may not be preserved after the transformation. A packet arriving earlier at a light-traffic station than a packet arriving at a heavy-traffic station may be transformed into a larger contention parameter than that at the heavy traffic station. Due to this phenomenon, the first-come-first-serve discipline proposed by Kurose and Schwartz [KUR83] cannot be applied on the transformed contention parameters.

(c) *Global Priority-Resolution Protocol*:

The optimization of window protocols for priority resolution is similar to but more efficient than protocols with continuous distributions because the contention parameters can only be assigned integers and recursion can stop when the window size is smaller than one. It follows that:

$$n(a,b) = 0 \qquad \text{for all } b-a \le 1 \quad (3.10)$$

These boundary conditions assure that contention can always be resolved in finite steps. The performance was evaluated by assuming that a packet has equal probability of being at any priority level. The results plotted in Figure 2 show that the expected number of contention steps is bounded by a constant regardless of channel load. The performance is improved when the channel load is heavy because there is increased certainty for a high priority message to exist. However, the performance is slightly worse when the number of priority levels is large.

(d) *Tree-Walk and Urn Protocols*:

The discrete distributions for both Tree-Walk and Urn Protocols are non-identical. Assuming that the origin of the window is at station 1, there are two properties that can be used to reduce the complexities of Eq's 3.3 to 3.5: (i) $F_i(k) = 0$ for $k < i$; and (ii) $F_i(a) = F_i(b)$ for $i \le a,b \le N$. From these, we obtain:

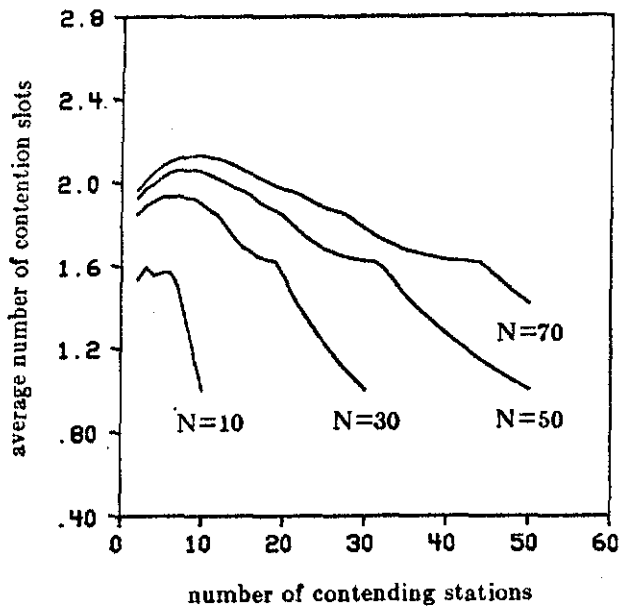

Figure 2. Performance of Adaptive-Tree-Walk and Urn Protocols optimized under dynamic-programming window control (N is the number of stations in the system).

$$Pr(A) = \prod_{i=1}^{a}[1-F_i(a)] - \sum_{i=a}^{b}F_i(b)\prod_{\substack{j=1\\ j\neq i}}^{b}[1-F_j(b)] - \prod_{i=1}^{b}[1-F_i(b)] \quad (3.11)$$

$$g(w,a,b) = \frac{1}{Pr(A)}\sum_{i=a}^{w}F_i(w) * \left\{\prod_{\substack{j=1\\ j\neq i}}^{w}[1-F_j(w)] - \prod_{\substack{j=1\\ j\neq i}}^{b}[1-F_j(b)]\right\} \quad (3.12)$$

$$r(w,a,b) = \frac{1}{Pr(A)}\left\{\prod_{i=1}^{w}[1-F_i(w)] - \sum_{i=w}^{b}\left[F_i(b)\prod_{\substack{j=1\\ j\neq i}}^{b}[1-F_j(b)]\right] - \prod_{i=1}^{b}[1-F_i(b)]\right\} \quad (3.13)$$

Among the protocols we have considered, the Tree-walk and Urn Protocols have the highest certainty about the values of x_i's. The performance is, therefore, expected to be the best. Figure 3 verifies this fact and shows that perfect scheduling can be achieved when the load is heavy. It should be noted that the performance degrades as the total number of stations increases. When $N \to \infty$, the protocol behaves like one with continuous distributions.

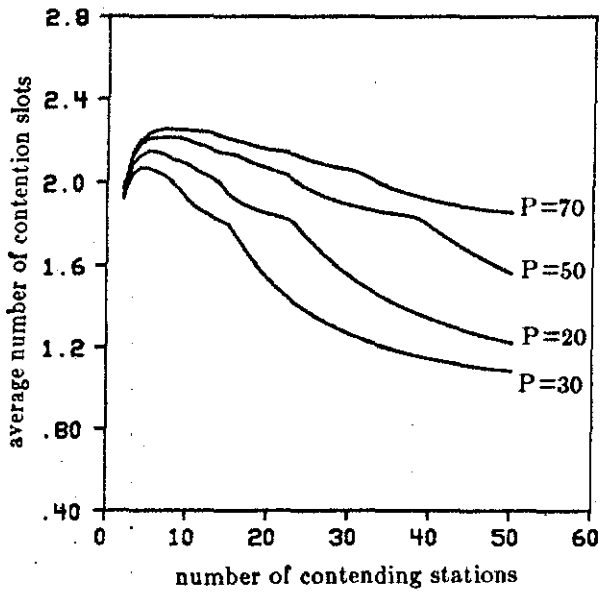

Figure 3. Performance of window protocol for global priority resolution operated under dynamic-programming window control (P is the number of priority levels).

4. IMPLEMENTATION CONSIDERATIONS

The dynamic-programming algorithm discussed in the last section provides a lower bound on the number of contention slots. However, the computational complexity is high which makes the algorithm impractical for real-time applications. As an example, when N=20 in the Tree-Walk or Urn Protocols, the execution time required on a VAX 11/780 to evaluate Eq. 3.2 is 1.3 seconds. When N=100, the time is increased to 828 seconds. In this section, approximate algorithms are proposed to evaluate window sizes efficiently. Further, the problem of estimating channel load will be addressed.

4.1. Approximation Algorithms

(a) *Binary Decision Trees:*

Given a channel load n, the sequence of optimal windows derived from Eq. 3.2 constitute a binary decision tree. The root of a subtree represents a window. The optimal window for the next slot will reside in the left subtree if collision is detected in the current slot. On the other hand, the optimal window for the next step is 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 optimal window in each contention step can, therefore, be retrieved efficiently. An assigned station will be responsible for updating the trees when distributions change. One problem with this method lies in the large memory-space requirement. Since the average number of contention slots is small, some subtrees can be pruned to reduce the memory space without significant degradation of performance. Window sizes in the pruned subtrees have to be obtained by interpolation techniques. Likewise, for those channel loads for which no decision trees are stored, interpolation has to be used to obtain window sizes.

(b) *Greedy Algorithms:*

To speed up processing, recursion can be restricted or eliminated in the dynamic-programming algorithm, and some local optimization is performed when the recursion terminates. In case that the contention parameters have identical continuous distributions F(x), using a window which maximizes the probability of success, g(w,a,b), in each contention step was found to be a good heuristic scheme. g(w,a,b) can be expressed in a simple form here:

$$g(w,a,b) = K\,[F(w)-F(a)]\left\{[1-F(w)]^{n-1}-[1-F(b)]^{n-1}\right\} \quad (4.1)$$

where $K = n/P\{A\}$. It can be shown that Eq. 4.1 is unimodal between a and b, so a maximum exists in the interval (a,b). To find the optimal value of w, we set $\frac{d}{dw}[g(w,a,b)] = 0$ and solve for w. Assuming that $f(w)\neq 0$, this leads to following equation:

$$[1-F(w)]^{n-1} - [1-F(b)]^{n-1} = (n-1)[F(w)-F(a)][1-F(w)]^{n-2} \quad (4.2)$$

Let $z = 1-F(w)$, Eq. 4.2 becomes:

$$z^{n-1} - \frac{(n-1)[1-F(a)]z^{n-2}}{n} - \frac{[1-F(b)]^{n-1}}{n} = 0 \quad (4.3)$$

It can be shown that a real root of Eq. 4.3 exists and satisfies $(1-F(b)) < z_o < (1-F(a))$. The optimal window w_o can be computed directly from z_o as follows:

$$w_o = F^{-1}(1-z_o) \quad (4.4)$$

There is no closed-form solution to Eq. 4.4. Although z_o can be solved numerically, it is still not practical for real-time applications. We derive an approximate solution to Eq. 4.1 by solving the following equation.

$$g(w,a,b) = K\,[F(w)-F(a)]\,[F(b)-F(w)] * [1-F(w)]^{n-2} \sum_{i=0}^{n-2} v^i \quad (4.5)$$

where $v = [1-F(b)]/[1-F(w)]$. An approximation function $\hat{g}(w,a,b)$ can be solved by substituting the term $\left[\sum_{i=0}^{n-2} v^i\right]$ by $(n-1)$. That is,

$$\hat{g}(w,a,b) = K'\,[F(w)-F(a)]\,[F(b)-F(w)]\,[1-F(w)]^{n-2} \quad (4.6)$$

where $K' = (n-1)K$. $\hat{g}(w,a,b)$ has its maximum at a position very close to that of g(w,a,b) and can be obtained by solving $\frac{d}{dw}[\log \hat{g}(w,a,b)] = 0$. From this we obtain:

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

or equivalently,

$$[F(w)]^2 + C[F(w)] + D = 0, \quad (4.7)$$

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 Eq. 4.7 in the interval (F(a),F(b)) is given by

$$F(w_a) = \frac{-C-\sqrt{C^2-4D}}{2} \quad (4.8)$$

The approximate window, w_a, can then be calculated easily. Figure 4 shows that empirically the approximate window-control rule (Eq. 4.8) performs nearly as good as the optimal one (Eq. 4.4). The number of contention

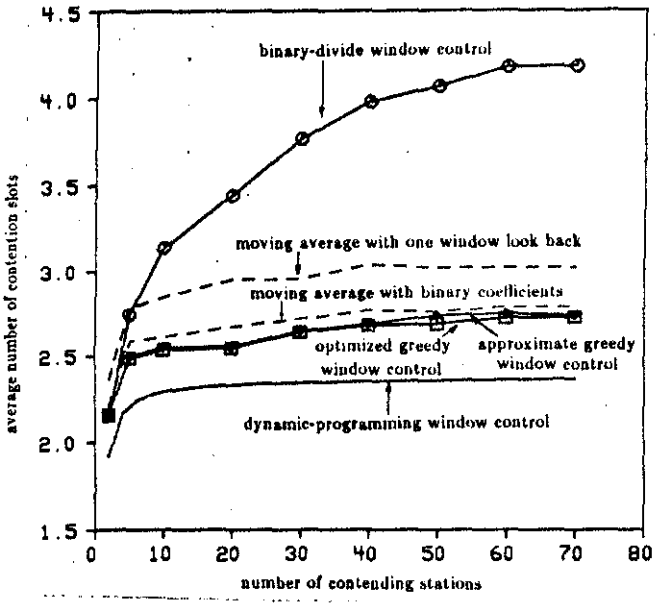

Figure 4. Performance of Virtual-Window Protocol (distribution of contention parameter is U(0,1)).

slots is bounded by 2.7 independent of the number of contending stations. Analytical verification of this bound is not shown here. It should be noted that Ethernet requires an average of $O(\log_2 n)$ contention slots [MET76], and the performance is shown approximately by the curve with binary-divide control rule in Figure 4.

When the distributions are discrete (such as in the priority resolution), maximization of g(w,a,b) has to be done by enumeration and not by differentiation. Approximations for applications with non-identical distributions (such as the Urn Protocol) will be investigated in the future.

4.2 Channel-Load Estimation

One of the key factors in optimizing window control is the information on channel load. This must be estimated directly from statistics collected in the network.

In the Virtual-Window Protocol the load information can be assessed easily. When contention ends, every processor knows the final window (a,w]. A maximum-likelihood estimate can be computed from the probability of success. The likelihood function is derived as:

$$L(n,w,a) = Pr(a<Y_1<w<Y_2) = n(w-a)(1-w)^{n-1} \quad (4.9)$$

L(n,w,a) is maximized at

$$n = \left\lfloor \frac{1}{\ln(1-w)} \right\rfloor \text{ or } \left\lceil \frac{-1}{\ln(1-w)} \right\rceil \quad (4.10)$$

Since the first-order statistic is readily available and can be 'piggybacked' in the packet transmitted, an alternative estimate is based on the density function of this statistic. The conditional density of y_1 is derived as:

$$f_{Y_1}(y \mid a<Y_1<w<Y_2) = \frac{\int_w^1 f_{Y_1Y_2}(y_1,y_2)dy_2}{\int_a^w \int_w^1 f_{Y_1Y_2}(y_1,y_2)dy_2 dy_1} \quad (4.11)$$

In the Virtual-Window Protocol, the distributions of the contention parameters are independent and uniformly distributed in (0,1), and we have:

$$f_{Y_1Y_2}(y_1,y_2) = n(n-1)(1-y_2)^{n-2} \quad (4.12)$$

Substituting Eq. 4.12 into Eq. 4.11 gives:

$$f_{Y_1}(y \mid a<Y_1<w<Y_2) = \frac{1}{w-a} \quad (4.13)$$

This result shows that the distribution of y_1 is determined once the final window is known. Therefore, no new information is gained by using the first-order statistic in the likelihood function.

The accuracy of prediction can also be improved by techniques in time-series analysis. An Auto-Regressive-Moving-Average model can be used to obtain an adjusted window w_{mv} over time. A simple example is:

$$w_{mv}(t) = (w_{mv}(t-1) + w) / 2. \quad (4.14)$$

In this model the influence of previous windows is reduced by a factor of two each time.

The performance of the Virtual-Window Protocol with estimated load is depicted in Figure 4. The average number of contention slots is 3.1 when n is estimated using the previous window alone, and the performance is very close to the optimum when n is estimated using moving averages.

For the Tree-Walk and URN Protocols, channel load can be characterized by the number of packets transmitted when the window has circumscribed around the stations once. This number can be collected directly on the network. For the Priority-Resolution Protocol, n can be estimated from the number of stations contending in the highest priority level. The analysis is similar and will not be shown here.

4.3 Optimization With Global Distributions

The dynamic-programming formulation relies on knowledge of channel load and distributions of contention parameters. If the distributions are known, it is possible to infer the load information from the window size. This explains why the Virtual-Window Protocol performs well with estimated channel load.

On the other hand, when the distributions are not available globally, the distribution of the first-order statistic can be estimated from contention activities on the network. However, the success probability, g(w,a,b), which depends on distributions of first-order and second-order statistics, cannot be derived from this information alone. It is only possible to predict the probabilities of collision and no transmission, thus the dynamic-programming formulation is reduced to:

$$n(a,b) = \min_{a<w<b} \left\{ \begin{matrix} 1 + n(a,w)\cdot\ell(w,a,b) \\ + n(w,b)\cdot r(w,a,b) \end{matrix} \right\} \quad (4.15)$$

such that

$$\ell(w,a,b) + r(w,a,b) = 1$$

Using this formulation, the overhead of contentions can no longer be kept constant and is increased to $O(\log_2 N)$. However, the performance depends on the entropy of the estimated distribution. A small entropy implies less uncertainty in identifying the minimum, and thus a better performance. This problem has been studied with respect to the resolution of priorities, and the entropy was found to be reduced when traffic is of bursty type. This issue will be covered in the future paper.

5. CONCLUSION

In this paper we have described a window-search scheme to find the extremum among a set of random numbers. This scheme can be implemented effectively on

CSMA/CD networks. It unifies a class of adaptive CSMA protocols and allows the optimization to be done by a unique method. Dynamic-programming formulation to minimize the expected total number of contention slots was studied and evaluated. The formulation was based on information on channel load and distributions of contention parameters. It was found that the average number of contention slots expended before the extremum was identified was bounded by a constant. This performance can be said to be optimal as far as the order of magnitude is concerned. In practice, channel load cannot be obtained directly and has to be estimated from the window size, the first-order statistic and the distributions of contention parameters. Future research lies in the investigation of contention parameters with less uncertainity in their values so that contention can be resolved in a shorter time.

REFERENCES

[CAP79] Capetanakis, J., "Tree Algorithm for Packet Broadcast Channels," *IEEE Trans. Information*, Vol. IT-25, No. 5, Sept. 1979, pp.505-515.

[FAY77] Fayolle, G., et al., "Stability and Optimal Control of the Packet Switching Broadcast Channel," *JACM*, Vol. 24, No. 3, July 1977, pp. 375-386.

[KLE75a] Kleinrock, L. and F. A. Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple Access Modes and Their Throughput- delay characteristics," *IEEE Trans. Communications*, Vol. COM-23, Dec. 1975, pp. 1400-1416.

[KLE75b] Kleinrock, L., *Queueing Systems, Vol. 1: Theory*, John Wiley, New York, 1975.

[KLE78] Kleinrock, L. and Y. Yemini, "An Optimal Adaptive Scheme for Multiple Access Broadcast Communication," *Proc. ICC*, pp. 7.2.1-7.2.5, 1978.

[KUR83] Kurose, J. F., and M. Schwartz, "A Family of Window Protocols for Time Constrained Applications in CSMA Networks," *Proc. IEEE INFOCOM 83*, San Diego, CA, 1983, pp. 405-413.

[MET76] Metcalfe, R.M., and D.R. Boggs, "Ethernet : Distributed Packet Switching for Local Computer Networks," *CACM*, Vol.19, July 1976, pp.395-404.

[MIT81] Mittal, K. and A. Venetsanopoulos, "On the Dynamic Control of the Urn Scheme for Multiple Access Broadcast Communication Systems," *IEEE Trans. Communications*, Vol. COM-29, July 1981, pp. 962-970.

[NI83] Ni, L. M. and X. Li, "Prioritizing Packet Transmission in Local Multiaccess Networks," *Proc. of Eighth Data Communications Symposium*, Cape Cod, MA, 1983.

[PAP65] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, N.Y. 1965.

[SHA83] Shacham, N., "A Protocol for Preferred Access in Packet-Switching Radio Networks," *IEEE Trans. Communications*, Vol. COM-31, No. 2, Feb. 1983, pp. 253-264.

[TOB82] Tobagi, F. A., "Carrier Sense Multiple Access with Message-Based Priority Functions," *IEEE Trans. Communications*, Vol. COM-30, No. 1, January 1982.

[TOW82] Towsley, D. and Venkatesh, G., "Window Random Access Protocols for Local Computer Networks," *IEEE Trans. Computers*, Vol. C-31, No. 8, August 1982, pp. 715-722

[WAH83] Wah, B. W. and J. Y. Juang, "An Efficient Protocol for Load Balancing on CSMA/CD Networks," *Proc. 8th Conference on Local Computer Networks*, October 1983, pp. 55-61.