A Queueing Theoretic Approach to the Delay Analysis for the FCFS 0.487 Conflict Resolution Algorithm

George C. Polyzos, *Member, IEEE,* and Mart L. Molle, *Member, IEEE*

Abstract—**We apply our queueing theoretic delay analysis methodology to several variations of the FCFS 0.487 Conflict Resolution Algorithm. In our approach, the main component of the packet delay is viewed as a queueing problem where each window selected by the channel access algorithm is a customer requiring conflict resolution as its service. In the past, we have shown that the method can give exact expressions for the steady-state *distribution* of the packet delay for other window access protocols satisfying a separability condition. This condition, not satisfied by the FCFS 0.487 Algorithm, requires the algorithm to resolve each conflict completely, without leaving any part of the associated window unexamined, so that service to one customer in our queueing model does not induce regression in subsequent arrival times. We now extend our methodology to handle all the features of the full FCFS 0.487 Algorithm, including variable-size windows, arrival-time addressing (and hence true FCFS scheduling), and biased interval splitting. Some of these extensions involve approximations, but they allow us to obtain the Laplace transform and moments of the packet delay. We also present an exact analysis for the Three-Cell Algorithm, which is the 0.487 Algorithm with a few modifications to satisfy our separability condition that reduce its capacity to 0.48. Comparisons made with other analyses (which provide bounds on the mean delay) and with extensive simulations show that our results for both the mean and variance of the packet delay are extremely accurate.**

Index Terms—**Random access, collision channel, tree algorithms, delay distribution, decomposition method.**

I. Introduction

CONFLICT resolution algorithms (CRA's) allow a set of stations to communicate over a common channel under distributed control.[1] These algorithms rely on group testing[2] to resolve scheduling conflicts among the stations. The actions of such an algorithm consist of a sequence of "steps" at which access permission is granted to some subset of the participants, as determined by the feedback at the present step together with the current state.[3] Because the feedback is assumed available to all participants,[4] they are able to keep a consistent view of the channel history. Thus, all participants can run individual copies of the algorithm to create a distributed access control scheme.

Although CRA's have been studied extensively, the novelty in the present work lies in the following areas. First, unlike most of the literature on the subject (which concentrates on capacity and/or stability analysis), our results are in the area of delay analysis. These results allow us to find the steady-state *distribution* of the packet delay exactly for a number of interesting configurations. Second, unlike other work in the area of delay analysis (e.g., [34], [5], [17], [12], [13], [7], [18], [8], [9], and [10]), we derive these results by modeling the channel access algorithm as a queueing system, which both increases the intuitive understanding of the mechanics of the system and puts at our disposal the results and the methods of the vast queueing theoretic literature. And finally, these results both complement and extend our study of finite population systems in [25], which focused on the Standard Tree Algorithm (STA) with deterministic addressing in combination with the Simplified Window Algorithm. That is, we now focus on the infinite population Poisson model and concentrate on the nonsimplified version of the Window Algorithm.

In the remainder of this section, we discuss the model, the STA, and the channel access algorithms, and we present our assumptions. In Section II, we present a summary of our delay analysis methodology for "separable" protocols, i.e., those that have a conflict resolution algorithm which completely resolves the set (of potential) contenders, as supplied by the channel access algorithm, before terminating. In Section III, we introduce the

Manuscript received September 1, 1991; revised September 22, 1992. This work was supported in part by the Regents of the University of California through a Chancellor's Faculty Fellowship and by the Natural Sciences and Engineering Research Council of Canada under Grant A5517.

G. C. Polyzos is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093.

M. L. Molle is with the Computer Systems Research Institute, University of Toronto, Toronto, Ont., Canada M5S 1A4.

IEEE Log Number 9213106.

[1] Various CRA's have been introduced in [11], [32], [3], [16], [6], [33], [21], [36], and [23]. The book by Bertsekas and Gallager [1] provides a very good introduction into this field. The March 1985 special issue of the IEEE Transactions on Information Theory [31] includes many important contributions and perspectives to the field. In particular, [35] provides an excellent survey of Russian contributions to the area.

[2] See [37] for an introduction to group testing and its relationship to CRA's.

[3] The execution of many of these algorithms may be conveniently viewed as the traversal of a tree, hence the name "Tree Algorithms."

[4] We concentrate on full sensing CRA's in this paper.

Three-Cell Algorithm (3CA) and analyze its performance. This algorithm is a simplified version of the FCFS 0.487 Algorithm with capacity equal to 0.48, that retains almost all the essential features of the original algorithm, but does not induce regression in time.

Finally, in Section IV, we analyze the full FCFS 0.487 Algorithm with addressing (splitting) based on packet arrival times—giving a true FCFS system. This protocol uses "tree pruning" and induces regression in time. In our analysis, we introduce a new transformation that counteracts the effects of the regression of the window in the 0.487 Algorithm, thereby making all *transformed* windows *within* a busy period to be of constant length. This transformation greatly simplifies the problem of obtaining the statistics of the lag by using our queueing theoretic methodology since it decouples the service time for a customer (i.e., the transformed epoch length) in the corresponding queueing system from the interarrival time up to the next customer (i.e., the transformed window size). In solving this system, we make the approximation that the transformed window sizes are bimodal (i.e., either one slot or w slots, and never any intermediate values). This, together with an approximation we introduce in the solution of the queueing system, means that our results are not exact for the 0.487 Algorithm. Nevertheless, by comparing our results with bounds on the mean delay obtained via other techniques and with extensive simulations, we show that our method is extremely accurate in terms of the mean and variance of the packet delay.

A. The Model

We start with the usual assumptions about the environment. There is a single, error-free, synchronous (slotted), broadcast channel. The packets are of fixed length, equal to the slot size, which is taken as the time unit. For simplicity, and easy comparisons with previous results, we neglect propagation delays and assume that errorless feedback is available to all participants at the end of each slot.[5] We assume a large number of independent stations, modeled as an infinite population, generating packets according to a Poisson process with aggregate rate λ (packets per slot).[6] Thus, with a window of size w, the number of packets in a window is Poisson distributed with parameter $x = \lambda w$. We denote by $\Phi(x)$, $\Psi(x)$, and $F(x)$ the probabilities of idle, success, and no-conflict. That is,

$$\Phi(x) = e^{-x}, \qquad \Psi(x) = xe^{-x}, \quad \text{and}$$

$$F(x) = \Phi(x) + \Psi(x) = (1 + x)e^{-x}.$$

[5]Among the CRA's considered in this paper, the Standard Tree Algorithm requires at least binary feedback of the type "conflict–no-conflict," while the Modified Tree Algorithm and all others using "level skipping," such as the Three-Cell Algorithm and the 0.487 Algorithm require at least ternary feedback of the type "idle–success–conflict." We consider channels with memoryless feedback errors in [28].

[6]This is the usual traffic model, adopted in most performance studies in the area. In [28], we show that a slightly more general model for the arrival process can be adopted, if desired.

In this paper, we focus on three CRA's. First, in Section II, we use the well-known binary, preorder traversal, Standard Tree Algorithm (STA), referred to as the Capetanakis CRA in [16]. Its simplicity allows us to illustrate the fundamentals of our method. Then, in Sections III and IV, we present and analyze the Three-Cell and the FCFS 0.487 Algorithms, respectively, introducing additional techniques to handle the new features in these algorithms. The STA operates as follows:

> If a conflict arises, the active set is partitioned into two subsets according to an addressing scheme. All conflicts in the first subset, if any, are resolved first, recursively, and then, those of the second subset, if any, are resolved (recursively).

The partitioning can be on the basis of random "coin tosses" (random addressing), or packet arrival times (arrival-time addressing), or on the basis of station addresses (deterministic addressing). It is well known from the work of Capetanakis [2] that in the finite population case, a deterministic addressing scheme performs better, and we have followed this approach in [25]. In the infinite population model, however, random and deterministic addressing become indistinguishable. Arrival-time addressing, on the other hand, leads to the same mean packet delay, but with reductions to the variance and higher moments of the delay, as would be expected since it enforces the FCFS rule [28]. We consider random addressing in Sections II and III, and treat arrival-time addressing in the case of the FCFS 0.487 Algorithm in Section IV.

In addition to a conflict resolution algorithm, a complete random access system must also have a channel access algorithm to specify when packets may join the conflict resolution algorithm. We consider two variations of the window algorithm, both of which fit the following general description:

> Assume that at time θ (measured in slots), the conflict resolution algorithm has just completed the transmission(s) of all packets that arrived before τ. Initially, we set $\theta \leftarrow 0$, $\tau \leftarrow 0$. Then, at time θ', the channel access algorithm permits all packets that arrived in the window $(\tau, \tau + w']$ to be transmitted. After all of these packets have been transmitted successfully (with the aid of the conflict resolution algorithm) using l slots ($l \geq 1$), we repeat the process after setting $\theta \leftarrow \theta' + l$ and $\tau \leftarrow \tau + w'$.

The two algorithms differ only in the specification of θ' and w'. The first, called the *Window Algorithm* (WA), specifies that

$$\theta' = 0 \quad \text{and} \quad w' = \min\{\theta - \tau, w\}, \qquad \text{(WA)}$$

i.e., windows of size w are formed, except when this is not possible because the time from the beginning of the window until the current time is less than w. In this case, a smaller window is chosen. The WA was first used by Gallager [6] in connection with the FCFS 0.487 Algorithm, although he did not explicitly mention the channel access scheme. Gallager's protocol also uses a

"tree pruning" inference rule, which sometimes leaves part of a window unresolved. To handle this case, the update to τ in the above algorithm description should be replaced by $\tau \leftarrow \tau - r + w'$ where r is the part of the previous window that remained unresolved.

The second, called the *Simplified Window Algorithm* (SWA), is a slightly modified version of the WA that simplifies the analysis, but has the essential characteristics of the Window Algorithm. Here,

$$\theta = \max\{\theta, \tau + w\} \quad \text{and} \quad w' = w, \qquad \text{(SWA)}$$

i.e., windows of size w are always selected, even in the cases when the difference between the beginning of the window and the current time is smaller than w, by introducing a "rest" period, i.e., delaying the decision until such time that would permit the formation of a window of size w. Massey [16] explicitly discussed the distinction between the WA and the SWA, and took advantage of the latter to simplify the capacity analysis of the STA—it is shown in [16], and will be obvious in later sections of this paper, that the WA and the SWA have the same capacity. An illustration of the operation of a random access protocol using the STA in combination with either the SWA or the WA is shown in Fig. 1.

Note that in the case of the SWA, the window size w' is *constant* ($=w$). This property is crucial for modeling part of the system as a $D/G/1$ queue; this was the approach we used in [25]. There is, however, an obvious improvement in delay if the WA is used since it uses the time that would have been wasted in a "rest" period to start an epoch from a "small" window. The difference in mean delay at light loads between the two algorithms is one or two slots for the most common values of the window size (i.e., $w = 2, 3$).

II. A Queueing Theoretic Delay Analysis Methodology

In this section, we briefly present our delay analysis methodology for "separable" protocols using random addressing. This methodology is then applied to the Three-Cell Algorithm in Section III, and is extended in order to analyze the FCFS 0.487 Algorithm in Section IV.

We define the random variable t to be the *total delay* experienced by a randomly selected packet, from its moment of generation at its source until the end of its successful reception at its destination. In order to find expressions for the total delay, we find it convenient to express t as the sum of three component random variables t_0, t_1, and t_2, which we shall evaluate using different techniques. In particular, we treat the operation of the CRA as a bulk-service queueing system, where all packets whose arrival times fall within the same arrival time window join the same bulk, and their collective service requires one epoch of service by the CRA. Thus, with reference to Fig. 1 and our earlier discussion of channel access algorithms, we consider a "tagged" packet that arrives at time η and departs at time η'. If $(\tau, \tau + w']$

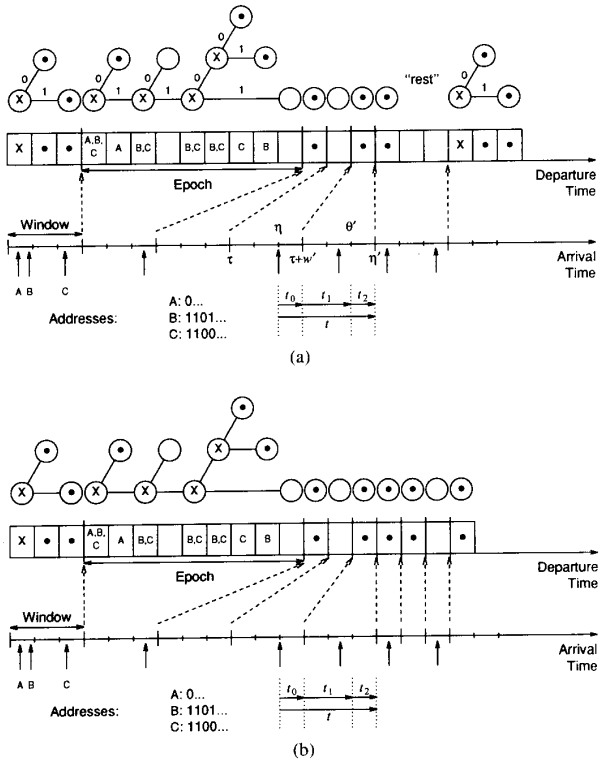

Fig. 1. Example of a random access system. Binary STA for conflict resolution and (a) SWA or (b) WA for channel access. The same scenario is presented to both protocols for comparison.

represents the window in which the "tagged" packet departs from the system, then we define

$$t_0 = \tau + w' - \eta$$

$$t_1 = \theta' - \tau - w'$$

$$t_2 = \eta' - \theta'$$

and, of course,

$$t \triangleq t_0 + t_1 + t_2 \equiv \eta' - \eta.$$

In other words, t_0 represents the *residual life* of the window generation process, t_1 represents the *lag* between the generation of a completed window and the start of its associated epoch, and t_2 represents the *age* of the epoch at the moment of departure for the "tagged" packet. Note also that all packets belonging to a particular window share a *common value* for t_1.

We devote most of this section (and, indeed, this paper) to describing how we can obtain the statistics of the lag t_1, which is the most important component of the delay. However, in order to achieve this, our queueing theoretic methodology requires knowledge of statistics of the epoch length of the CRA under consideration, which plays the role of the service time in our queueing model. We

present the necessary notation and briefly describe techniques for obtaining statistics of the epoch length in Section II-A below. Finally, in order to characterize the total delay, in addition to t_1, we need the statistics of the other two components of the delay t_0 and t_2.

The initial part of the delay t_0 is merely the residual life of the current window at the moment when a packet is generated. For example, with Poisson arrivals, the initial delay is uniformly distributed in $[0, w)$. Depending on whether we assume a continuous- or discrete-time model, we use either the Laplace transform of t_0, denoted by $\Theta^*(w, s)$, or the probability generating function (PGF) of t_0, denoted by $\Theta(w, z)$.

The PGF for the delay in the epoch of transmission, denoted by $G(x, z)$, can be obtained by solving a functional equation similar to that obtained for the epoch length. In Section III, we use the standard way of obtaining statistics for metrics such as the epoch length and the delay of a packet in the epoch of transmission, i.e., we condition on the number of packets in an epoch. Then, in Section IV, we use the functional equation approach again, but this time we obtain functional equations on statistics of the sum of t_0 and t_2 in order to deal with the dependence between them.

After obtaining the transforms of the distributions of the components of the delay for a (tagged) packet, we obtain the transform of the distribution of the total delay as their product. Notice that with random (and deterministic) addressing, the components of the delay, conditioned on a given window size, are independent random variables since

t_0: depends only on its relative position within the window of arrival, which is independent of all other processes in the system,

t_1: depends only on traffic generated before its window of arrival, and

t_2: depends only on the number of packet arrivals within the window of arrival.[7]

The final result (in the all discrete-time model), the PGF of the total delay $D(z)$ can then be obtained as the product of the PGF's of the three components of the delay in the case of the SWA (which uses constant size windows):

$$D(z) = \Theta(w, z)\, W(z)\, G(x, z).$$

If the continuous-time model is adopted for the initial delay, the Laplace transform of the total delay $D^*(s)$ can be obtained as

$$D^*(s) = \Theta^*(w, s)\, W(e^{-s})\, G(x, e^{-s}).$$

With the WA, where the window size varies, the above equations hold only when conditioned on a specific window size; therefore, an additional unconditioning step is required in that case (for example, see (6) in Section II-C).

A. Epoch Length and Capacity

One of the basic metrics that characterize the efficiency of a conflict resolution algorithm is the *epoch length*, i.e., the time required to resolve a conflict, which we denote by the random variable l. We are particularly interested in the epoch length distribution in the case of a set of initial contenders (i.e., packets), coming from a Poisson distribution with parameter x. A convenient way to handle epoch lengths is through their PGF, $Q(x, z)$, which is defined as

$$Q(x, z) \triangleq \sum_{n=0}^{\infty} \Pr\{l = n \mid x\} z^n.$$

Functional equations on $Q(x, z)$ and other epoch length statistics for various CRA's have been obtained and solved in many previous papers (for example, see [36], [17], [5], [26], [28], and others).

If only the mean delay is of interest, then only the first two moments of the epoch length are required. Therefore, instead of solving the functional equations in $Q(x, z)$, we may obtain and solve functional equations in the moments under consideration, which are functions of a single variable. We denote the first two moments of the epoch length as

$$L(x) \triangleq E[l \mid x] \quad \text{and} \quad H(x) \triangleq E[l^2 \mid x],$$

respectively.

Since we deal with an infinite population model and the separable CRA's guarantee that by the end of an epoch all packets involved in the original conflict have been successfully transmitted, the average throughput and input rate λ coincide as long as the system is stable. To find the capacity of the protocols, we have to consider the conditions under which there is no long-term backlog in the system.[8] Intuitively, it is easy to justify the following condition for stability:

$$L(w\lambda) < w. \tag{1}$$

This condition simply states that the system is stable as long as the mean time to resolve a conflict is less than the time between successive requests.[9] To be rigorous, we have to show that this condition is sufficient for enough moments of t to be finite. Assuming the above decomposition $t = t_0 + t_1 + t_2$, and observing that t_0 is bounded by w, and also that all the moments of t_2 are bounded for bounded x, we can identify stability with bounded moments of t_1. But as we will see from the queueing models developed below, when the condition of (1) holds, all the moments of the lag t_1 are finite.

[7] Arrival-time addressing, which is critical for the operation of the FCFS 0.487 Algorithm, introduces a dependence between t_0 and t_2. However, even in this case, the lag t_1 is independent of the sum of t_0 and t_2. We handle this complication in Section IV.

[8] The standard approach to obtain capacity results for CRA's is through the use of Pakes' lemma [22]. However, we believe that the alternate technique presented here is interesting and intuitive.

[9] This condition (or similar ones) were presented by several authors, e.g., [6], [36], [19], [24].

B. Constant Size Window Access—The Simplified Window Algorithm

We begin the analysis of the statistics of the lag with the SWA, in which $w' \equiv w$ for every epoch. Even though this case is of less practical interest than the WA case analyzed next, it serves as an introduction to our later models and transformations, making them more natural and helping intuitive understanding.

1) Modeling t_1 as a $D/G/1$ Queueing System: Since all packets belonging to the same window share a common value for t_1, and under the SWA the number of packets found in each window are i.i.d. random variables, it makes no difference whether we evaluate the statistics of t_1 in terms of what is "seen" by a randomly chosen individual packet or by a bulk arrival of packets belonging to a randomly chosen window. Therefore, we look at the system at the window level, and model it as a queueing system. We regard all windows as customers that require service in the form of conflict resolution. Service time is then, clearly, the length of the epoch generated by the window, with known (steady-state) distribution $B(z)$, which is given by $Q(x, z)$ under the Poisson arrival model. Because of our assumption of constant window size, the arrival process is deterministic and service times are i.i.d. Thus, if only integer values are assumed for the window size (and since this is clearly the case for the epoch lengths), the system can be modeled as a *discrete-time $D/G/1$ queue*. We have presented this approach to the delay analysis in [25]. Some additional treatment required for the case of rational window sizes is presented in [29].

An analysis of the discrete time $D/G/1$ queue can be found in [30], and generally involves the numerical computation of the complex roots of a polynomial in the unit circle. When many roots have to be found numerically, the suggested approach is through an iterative search around values of the form $\rho e^{j\theta}$ with $\rho \approx 1$, $\theta \approx 2\pi n/(w-1)$, and $n = 0, 1, \cdots, w-2$ (see [30] for more details).

2) Modeling t_1 as a Moving Server $M/G/1$ Queueing System: Let us take another look at the calculation of the lag, i.e., t_1, for the system we considered above. This time, however, in contrast to the $D/G/1$ queue of the previous section, we now use a different approach that can be extended to accommodate the (nonsimplified) Window Algorithm (WA). The approach we are adopting here is a special case of the "Moving Server" technique [19], [20].

We first need to make the following observation. Consider the standard, general, queueing system equation (see, for example, [4])

$$\omega_{n+1} = \max\{\omega_n + l_n - \alpha_{n+1}, 0\} \tag{2}$$

where ω_n is the waiting time for the nth customer, l_n is the service time for the nth customer (the length of the nth epoch in our case), and α_{n+1} is the interarrival time between the nth and the $(n+1)$st customers. Observe that there is no effect on the waiting times if both the service times and the interarrival times are each decreased by one unit (for all customers). This observation permits us to make our first transformation (leaving the waiting times, as random variables, unaffected):

$$\tilde{l} = l - 1 \quad \text{and} \quad \tilde{w} = w - 1.$$

Note that l is the random variable for the epoch length, and that the interarrival time is always w, i.e., equal to the window size. The service-time distribution is now

$$\frac{Q(x, z)}{z}.$$

As a result of this transformation, some customers of our (transformed) queueing system require exactly zero service time. These are windows that result in no-conflict, with probability $\bar{p} = F(x) \triangleq (1+x)e^{-x}$. Since these "customers" do not require any service, we can just ignore them and consider as customers only conflict windows. Notice that a window is a conflict window with probability $p = \bar{F}(x)$, independent of all other processes in our system.[10] This last change of viewpoint changes the service-time distribution for our system to

$$\tilde{B}(z) \triangleq \frac{Q^{(c)}(x, z)}{z} = \frac{Q(x, z) - zF(x)}{z\bar{F}(x)}.$$

What we have now is the following situation. Every $w-1$ slots, we either get a customer arrival (i.e., a conflict window) or not, with respective probabilities p and $\bar{p}$. In other words, we have a memoryless geometric interarrival time distribution in discrete time. Furthermore, the service times for these customers are i.i.d. (and also independent of the operation of the system) with (discrete) distribution whose PGF is $\tilde{B}(z)$. Thus, if we are lucky, our transformed system can be modeled as a discrete time $M/G/1$ queue, with the transformed window (i.e., $w-1$ slots) as its elementary time unit. Therefore, we are led to the following transformation:

$$\tilde{\tilde{l}} \triangleq \frac{\tilde{l}}{w-1} = \frac{l-1}{w-1} \quad \text{and} \quad \tilde{\tilde{w}} \triangleq \frac{\tilde{w}}{w-1} = \frac{w-1}{w-1} = 1.$$

From the point of view of the service-time distribution, this last transformation can be expressed as

$$B(z) = \tilde{B}(\zeta)|_{\zeta^{w-1}=z}.$$

For this last transformation to be possible (i.e., exact), we must have

$$(l-1) \bmod (w-1) = 0. \tag{3}$$

This last condition is obviously satisfied for $w = 2$. In addition, for the binary STA, the condition is satisfied for $w = 3$ as well since epoch lengths are odd integers.[11] Furthermore, if we attempt to apply the results of this technique even when (3) is not satisfied, we discover that

[10] We use the notation $\bar{f} \triangleq 1 - f$.

[11] This condition is satisfied for many more interesting system parameters when one considers more general environments, where the duration of idle, success, and conflict slots might be different [27], [28].

we obtain extremely good approximations to the true metrics, which can be obtained through the $D/G/1$ analysis [28]. Therefore, this is a viable technique, and its practicality is not limited to a few interesting window sizes. The reason for this serendipitous success is discussed below.

With this last transformation, we have obtained a true discrete-time $M/G/1$ queue. We are interested in the distribution of the waiting time in such a system, which can be easily obtained (for example, see [14]). This distribution can be put in the form of

$$\Omega(z) = \frac{R(1)}{R(z)}, \qquad R(z) = 1 - p\frac{B(z) - 1}{z - 1}$$

where $B(z)$ is the service-time distribution and p is the arrival probability. The inverse transformation only requires that we reinstate the old unit of time. The final result can then be obtained as

$$W(z) = \Omega(z^{w-1}) = \frac{R(1)}{R(z^{w-1})}$$

with

$$\begin{aligned} R(z^{w-1}) &= 1 - p\frac{B(z^{w-1}) - 1}{z^{w-1} - 1} \\ &= 1 - \bar{F}(x)\frac{\dfrac{Q(x,z) - zF(x)}{z\bar{F}(x)}}{z^{w-1} - 1} \\ &= 1 - \frac{\dfrac{1}{z}[Q(x,z) - zF(x)] - \bar{F}(x)}{z^{w-1} - 1} \\ &= 1 - \frac{Q(x,z) - z}{z^w - z} = \frac{z^w - Q(x,z)}{z^w - z}. \end{aligned}$$

Thus, with both the $D/G/1$ model [25] and the "Moving Server" $M/G/1$ model, the waiting time distribution is of the form

$$W(z) = \frac{R(1)}{R(z)}$$

with

$$R(z) = \begin{cases} \dfrac{z^w - Q(x,z)}{z\prod_{i=1}^{w-1}(z - z_i)}, \\ \dfrac{z^w - Q(x,z)}{z(z^{w-1} - 1)} = \dfrac{z^w - Q(x,z)}{z\prod_{i=1}^{w-1}(z - e_i)}, & \text{"M.S."} \\ & M/G/1. \end{cases}$$

Notice that the two expressions for $R(z)$ differ only in the way the roots of the denominator are defined. Where the $D/G/1$ expression uses the $w - 1$ nonzero roots of $z^w - Q(x, z)$ that are inside or on the unit circle, the "M.S." $M/G/1$ expression uses the roots of unity of order $w - 1$. As we have mentioned, the second expression approximates the first one very well in cases where it is not exact. The reason is that the roots required in the first expression are close to the $(w - 1)$st roots of unity. This is the case exactly when $x \to 0$, but even for values of x close to the capacity of the algorithms, the difference is small [28].

C. Window Access with No "Rest" Periods—The Window Algorithm

Our technique of modeling t_1 as the waiting time in a discrete-time $M/G/1$ queue can be extended to accommodate the WA as well. The WA and the SWA differ only when there is no backlog in the system, where the WA selects a small window while the SWA insists on waiting (i.e., a "rest" period) until a window of the normal size can be selected. This is exactly the setting of the "generalized busy period" $M/G/1$ queue. It is an $M/G/1$ queue, with the only difference that the first customer in each busy period is different from the customers that come in the middle of a busy period, having different arrivals and service-time distributions.[12] The transformations we need to model the WA as a discrete-time "generalized busy period" $M/G/1$ queue are illustrated in Fig. 2.

An analysis of the discrete-time "generalized busy period" $M/G/1$ queue is provided in [28, Appendix] following the methodology presented for the continuous-time case in [15]. The conditional PGF for the waiting time for customers that arrive within (and hence do not initiate) a busy period is given by

$$W_b(\zeta) = \frac{1 - \rho}{\rho_0}\frac{p_0[B_0(\zeta) - 1]}{\zeta - 1 - p[B(\zeta) - 1]}$$

where $B_0(\zeta)$ and $B(\zeta)$ are the (transforms of the) distributions of service time and p_0 and p are the arrival probabilities for the first customer in a busy period and a "normal" one, respectively. The parameters ρ and ρ_0 are defined by $\rho_0 \triangleq p_0 B_0'(1)$ and $\rho \triangleq pB'(1)$, where $B_0'(1)$ and $B'(1)$ are the mean service times.

In our case, $p_0 = 1 - (1 + \lambda)e^{-\lambda}$, $p = 1 - (1 + w\lambda)e^{-w\lambda}$,

$$D/G/1 \qquad B_0(\zeta) = \left[\frac{Q^{(c)}(\lambda, z)}{z}\right]_{z^{w-1}=\zeta},$$

$$B(\zeta) = \left[\frac{Q^{(c)}(w\lambda, z)}{z}\right]_{z^{w-1}=\zeta}$$

[12] We should point out here that for the cases where we can get exact results, i.e., with $w = 2$ for all algorithms and $w = 3$ for the binary STA, there are only two categories of packets: those that arrived during a "small" window, of size 1, and those that arrived during a "large" window, of size w.

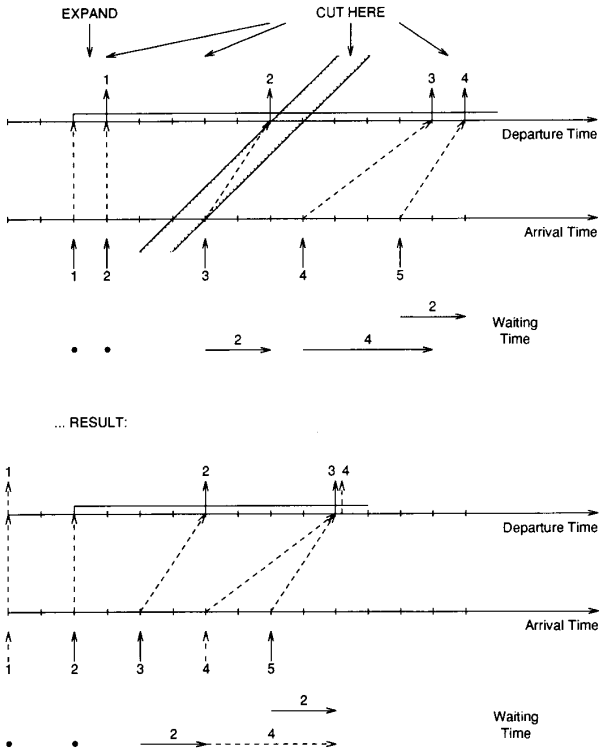

Fig. 2. The transformation to obtain t_1 as the waiting time in a generalized busy period, discrete time, $M/G/1$ queue.

and thus,

$$\rho_0 = \frac{L(\lambda) - 1}{w - 1} \quad \text{and} \quad \rho = \frac{L(w\lambda) - 1}{w - 1}. \tag{4}$$

The distribution of t_1 (for packets arriving in large windows) is obtained by changing the time unit back to slots:

$$W(w, z) = W_b(z^{w-1}).$$

The final result can be expressed as

$$W(w, z) = \frac{1 - \rho}{\rho_0} \frac{Q(\lambda, z) - z}{z^w - Q(w\lambda, z)}. \tag{5}$$

The PGF for t_1 needs to be combined with corresponding PGF's for t_0 and t_2 to obtain a PGF for t. If we focus on windows of a given size (large, say), the components of the delay are, again, independent random variables. However, all of them are correlated to the window size. In particular, $t_1 \equiv 0$ for small windows, and so is t_0 if we assume a discrete-time model of the arrival process. Thus, the PGF of the total delay[13] can then be obtained as the product of the PGF's of the three components of the delay for packets of large windows, and is just the PGF of t_2 for packets of small windows. The overall delay distribution is then given by

$$D(z) = \nu\Theta(w, z)W(z)G(\lambda w, z) + (1 - \nu)G(\lambda, z) \tag{6}$$

with ν the proportion of packets that arrive in large windows, which is obtained from the following equation:

$$\nu = \frac{\dfrac{B_0'(1)}{1 - \rho}w}{\dfrac{1}{p_0} + \dfrac{B_0'(1)}{1 - \rho}w} = \frac{w\rho_0}{1 - \rho + w\rho_0} = \frac{wL(\lambda) - w}{wL(\lambda) - L(\lambda w)} \tag{7}$$

where $B_0'(1)/(1 - \rho)$ can be recognized as the expected length of a generalized busy period and $1/p_0$ as the expected length of an idle period.

In Fig. 3, we plot the distribution of the packet delay for a system with the STA/WA combination with $w = 3$. Notice that the mass of the distribution (for light loads) is concentrated at $t = 1$, as desired, in contrast to the SWA case where the mass is distributed (equally at zero load) among $t = 1$, 2, and 3. We assume a discrete-time model for the initial delay here and in Fig. 4. Figure 4 shows the mean, standard deviation, and percentiles of the packet delay for the above system. The importance of the availability of percentile results for system design decisions should be obvious.

Finally, in Fig. 5, we provide cumulative distribution curves for various throughput values for the same protocol and parameters; these are more easily readable than the three-dimensional results of Fig. 3. Also shown are simulation results for the same system, except for a window size of 2.5 slots for comparison with [10, Fig. 3].[14] We observe strong agreement between our analysis and simulation, indicating the results are not very sensitive to the window size. Furthermore, it is interesting to note that $w = 3$ actually gives better results, except under very heavy load.

III. The Three-Cell Algorithm

The methodology described in Section II above applies directly to protocols with a conflict resolution algorithm which completely resolves all conflicts in a window. In that case, there is no overlap between the initial windows in successive epochs. We call the class of these windowed

[13] Since the existence of small windows has no effect on the boundedness of t_0 and the finiteness of the moments of t_2, once again, stability of the protocol is determined by t_1. As before, the moments of t_1 are finite, and the discrete-time queue is stable, whenever $\rho < 1$, which is equivalent to the stability condition of (1), as (4) shows.

[14] Because $w = 2.5$, we used a continuous-time model for the packet arrival process in the simulator. Thus, for compatibility, we incorporated an additional uniform (0, 1) term into our previous results for $w = 3$ to account for the initial delay distribution over a slot. We use simulation because the data from [10, Fig. 3] are both difficult to read and seem to be unreliable for $\lambda > 0.2$. We attribute the discrepancy to numerical problems encountered in the computation of the distribution in this last paper.

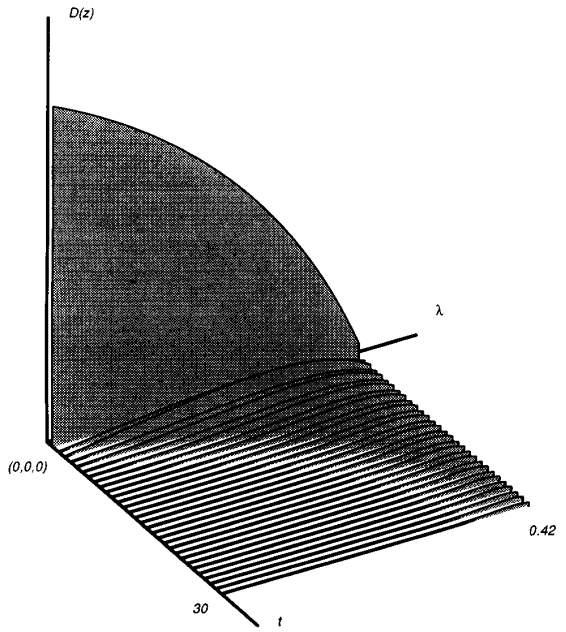

Fig. 3. The distribution of the delay for the WA ($w = 3$) with the binary STA.

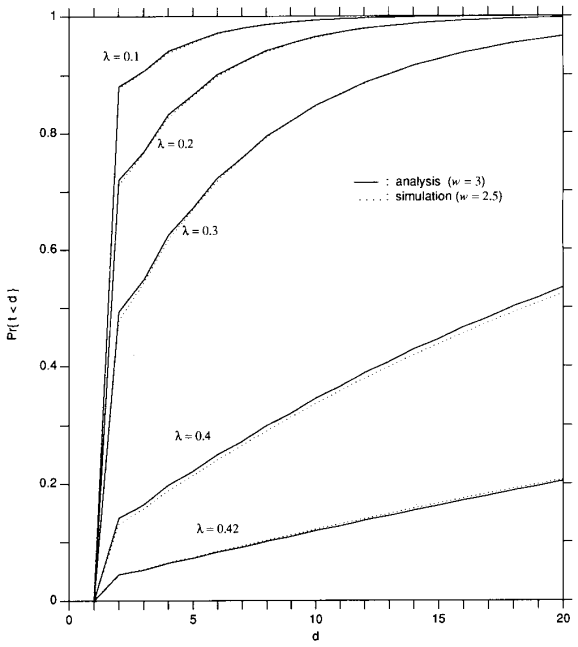

Fig. 5. Cumulative distribution of the packet delay for given throughput for the STA/WA (continuous-time model for the initial delay).

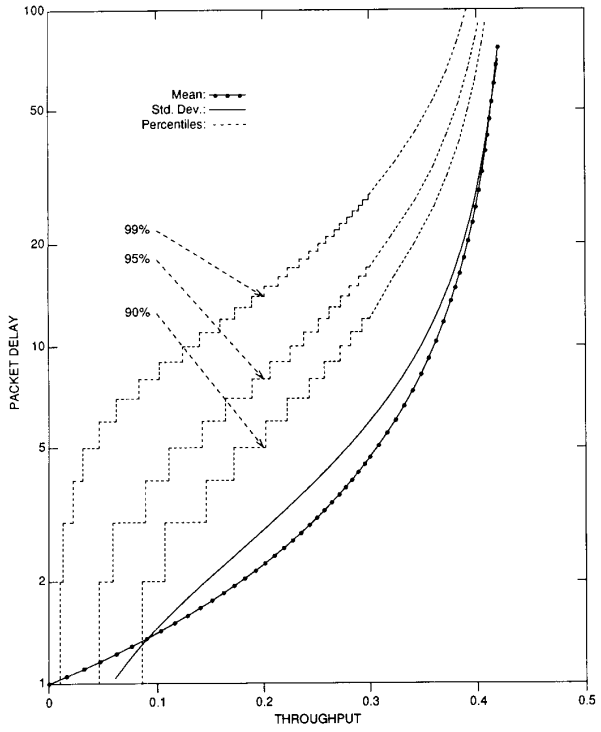

Fig. 4. Statistics of the packet delay for the binary STA with WA ($w = 3$). Random addressing, infinite population, Poisson arrivals, discrete-time model for the initial delay.

protocols *separable*.[15] In Section II, we have used the STA to illustrate our methodology. Here, we will introduce and analyze the Three-Cell Algorithm (3CA). This algorithm is quite similar to (but less complicated than) the limited stack depth algorithms introduced by Tsybakov and Likhanov in [36]. The most fundamental difference between the STA and the Modified Tree Algorithm (MTA) [16] on one hand, and the CRA in the FCFS 0.487 Algorithm on the other, is in the characteristics of their respective conflict resolution stacks. In the former algorithms, the set of contending users is partitioned using recursive binary (in general, d-ary) splitting until no element of the partition is found to contain more than one packet. Consequently, the stack depth increases by $d - 1$ after each conflict, and thus can potentially grow arbitrarily large. Conversely, the 0.487 Algorithm only retains information about those contending users that were included in the most recent previous conflict—all others are dropped from the current epoch by "regressing" the end of the window. As a result, it is well known that its stack depth never exceeds two cells.

In this section, we consider other algorithms with limited stack sizes. Unlike the 0.487 Algorithm, these algorithms work by combining (rather than eliminating) certain nodes in the conflict resolution tree used by the STA and the MTA. Thus, they retain our "separability" condi-

[15] The analogy with the class of separable queueing networks comes from the fact that the structure of the solutions to both classes can be factored into products of independent terms. However, the actual model solutions are unrelated.

tion of completely resolving all conflicts among the initial set of contenders. The Two-Cell Algorithm (2CA), introduced in [23], is a variation on the STA where only the top node on the conflict resolution stack is distinguished from the others. The resulting algorithm can be viewed as iteratively invoking a binary search to isolate the first packet from the remaining ones, until all have been transmitted successfully. The main attraction of the 2CA is its robustness; under window channel access, it attains about the same capacity as the STA on a noiseless channel, but as the channel error rates increase, it performs much better, achieving nonzero throughputs even for very high error rates.

The Three-Cell Algorithm (3CA), which we introduce in this paper, is a variation on the MTA where only the top two nodes on the stack are distinguished from the others. Since the nodes being accumulated in the third cell are exactly the ones that would be dropped from the current epoch by the 0.487 Algorithm, the resulting algorithm is almost identical to the 0.487 Algorithm. The only difference between the 3CA and the 0.487 Algorithm occurs at the point where the latter terminates after having dropped some of the nodes from the conflict resolution tree by regressing the window. (Recall that, under the Poisson traffic model, the dropped portion of the window is effectively unexamined.) Where the 0.487 Algorithm would attempt to augment the regressed window with newer arrivals, restoring its length to w before restarting the CRA (and thereby causing the initial windows in successive epochs to overlap, violating our separability condition), the 3CA is simply restarted on the regressed window. As a result of starting some epochs from a suboptimal window size, the 3CA is less efficient than the 0.487 Algorithm. Nevertheless, the loss in capacity is quite small (less than 0.5%), and the algorithm retains the essential features of the 0.487 Algorithm. Thus, since the 3CA also satisfies our separability conditions, it can be analyzed exactly by using our queueing theoretic methodology. The resulting performance estimates can be used to approximate the performance of the 0.487 Algorithm—but with the approximation taking place in the protocol specification instead of the model solution.

A. Epoch Length and Capacity

To apply our methodology, we begin by finding the PGF of the epoch length. In this case, it is more convenient to condition on the number of packets included in the interval under consideration because of the additional memory in the algorithm used to avoid the regression on the arrival time axis. Therefore, we define the following two families of conditional PGF's. First, we denote by $Q_n(z)$ the PGF of the epoch length of a "fresh" interval containing exactly n packets. Second, we denote by $Q_{n,m}(z)$ the PGF of a subepoch with a primary interval containing exactly n packets, but also including a secondary (right) interval including exactly m packets $(n, m = 0, 1, 2, \cdots)$. The subepoch is assumed complete after both the primary and secondary intervals are resolved completely.

From the definition of the algorithm, we obtain the following recursions for the PGF of the epoch length:

$n \le 1$:

$$Q_n(z) = z$$

$n > 1$:

$$Q_n(z) = zQ_n(z)B_{n,0} + z^2Q_{n-1}(z)B_{n,1} + z\sum_{j=2}^{n} Q_{j,n-j}(z)B_{n,j}$$

$$= \frac{z^2Q_{n-1}(z)B_{n,1} + z\sum_{j=2}^{n} Q_{j,n-j}(z)B_{n,j}}{1 - zB_{n,0}}$$

$n \le 1$:

$$Q_{n,m-n}(z) = zQ_{m-n}(z)$$

where

$n > 1$:

$$Q_{n,m-n}(z) = zQ_{n,m-n}(z)B_{n,0} + z^2Q_{n-1,m-n}(z)B_{n,1} + z\sum_{j=2}^{n} Q_{j,m-j}(z)B_{n,j}$$

$$= \frac{z^2Q_{n-1,m-n}(z)B_{n,1} + z\sum_{j=2}^{n-1} Q_{j,m-j}(z)B_{n,j}}{1 - zB_{n,0} - zB_{n,n}}$$

where

$$B_{n,j} \triangleq \binom{n}{j}\pi^j\bar{\pi}^{n-j},$$

i.e., the probability of j of the n packets ending up in the left subset, when each packet chooses randomly to join the left subset with probability π (or the right with probability $\bar{\pi}$). Note that the mean conditional epoch length for the 3CA is shorter than that of the MTA with biased splitting, but only for epochs containing six or fewer packets [29]. However, using the window algorithm for channel access makes sure that epochs with few packets are the most probable, and thus the 3CA has a higher capacity than the MTA. The unconditional PGF of the epoch length can then be obtained as

$$Q(x; z) = \sum_{n=0}^{\infty} Q_n(z)P_n(x)$$

where $P_n(x) = e^{-x}x^n/n!$ for Poisson arrivals.

From (1), we obtain 0.4799 as the capacity of the algorithm, with an optimal window size $w^* = 2.7$ and optimal splitting bias $\pi^* = 0.46$. With unbiased splitting, the maximum throughput is 0.478 (at the same w^*). With $w = 2$ and 3, the maximum throughput is 0.4674, and 0.4792, respectively (with $\pi = 0.46$).

B. Delay Analysis

Unfortunately, unlike in the case of the STA, the epoch length is not necessarily an odd integer. Thus, with the

$D/G/1$ model, we will in general require numerical computation of roots in order to get an exact expression for the distribution of the lag. Fortunately, such root-finding can be avoided completely for $w = 2$ (which leads to a capacity of 0.4674 with $\pi = 0.46$), and for $w = 3$, only the third root need be found numerically.[16] With $w = 3$, the capacity is 0.4792, i.e., within 0.16% of the optimum for this algorithm.

The last component needed is the delay in the epoch of transmission, which can be obtained in the same way we obtained the statistics of the epoch length. We denote by $G_n(z)$ and $G_{n,m}(z)$ the conditional PGF's for the delay in the epoch of transmission of a randomly selected packet analogously to the definitions of $Q_n(z)$ and $Q_{n,m}(z)$. Then, we obtain the following recursions:

$n=1$:

$$G_n(z) = z$$

$n>1$:

$$\begin{aligned} G_n(z) &= zG_n(z)B_{n,0} \\ &\quad + \left[\frac{1}{n}z^2 + \frac{n-1}{n}z^2G_{n-1}(z)\right]B_{n,1} \\ &\quad + z\sum_{j=2}^{n} G_{j,n-j}(z)B_{n,j} \\ &\quad \frac{z^2}{n}[1 + (n-1)G_{n-1}(z)]B_{n,1} \\ &\quad + z\sum_{j=2}^{n} G_{j,n-j}(z)B_{n,j} \\ &= \frac{}{1 - zB_{n,0}} \end{aligned}$$

$n=1$:

$$\begin{aligned} &G_{n,m-n}(z) \\ &\quad = \frac{1}{m}z + \frac{m-n}{m}zG_{m-n}(z) \end{aligned}$$

$n>1$:

$$\begin{aligned} &G_{n,m-n}(z) \\ &\quad = zG_{n,m-n}(z)B_{n,0} \\ &\qquad + \left[\frac{1}{m}z^2 + \frac{m-1}{m}z^2G_{n-1,m-n}(z)\right]B_{n,1} \\ &\qquad + z\sum_{j=2}^{n} G_{j,m-j}(z)B_{n,j} \\ &\qquad \frac{z^2}{m}[1 + (m-1)G_{n-1,m-n}(z)]B_{n,1} \\ &\qquad + z\sum_{j=2}^{n} G_{j,m-j}(z)B_{n,j} \\ &\quad = \frac{}{1 - z(B_{n,0} + B_{n,n})} \\ &\qquad z\sum_{j=2}^{n-1} G_{j,m-j}(z)B_{n,j} \\ &\qquad + \frac{}{1 - z(B_{n,0} + B_{n,n})}. \end{aligned}$$

[16] We do know, however, that this root will be real and less than or equal to one in absolute value, which makes the search easy. Furthermore, we can easily prove that this third root lies in $(-1, 0)$. We can get bounds on the mean delay that are *only half a slot apart* without even bothering to find this root [30], [29]. Furthermore, a rational window size of $w = 8/3$ leads to a capacity of 0.4798 when $\pi = 0.46$, and requires numerical computation of two pairs of complex conjugate roots (three more roots are equal to zero, and one is equal to one).

The unconditional PGF of the delay in the epoch of transmission can then be obtained. Differentiating, we obtain expressions for the mean $T_2(x)$ and the variance $\mathrm{Var}[t_2 \mid x]$ of the delay in the epoch of transmission.[17]

In Fig. 6, we present comparative performance results for the STA, the MTA, the optimally biased MTA (MTA*), the 3CA Algorithm, and the (unbiased) FCFS 0.487 Algorithm, all combined with the Simplified Window Algorithm (SWA) for channel access. All the results are analytical; the first four are exact, and the FCFS 0.487 curve is an extremely good approximation.[18] In Fig. 7, we provide results for the standard deviation of the packet delay for the same configurations and random addressing for the STA, MTA, and MTA*, and arrival-time addressing (as required) for the FCFS 0.487 Algorithm.

IV. The FCFS 0.487 Algorithm

We now turn our attention to the celebrated FCFS 0.487 Algorithm. Here, we have to address a number of additional complications compared to the 3CA. First, we have to deal with the regression on the arrival time axis. This regression occurs whenever a conflict is observed within the left subset of an earlier conflict. In this case, the 0.487 algorithm does not attempt to resolve the right subset (because no information about its contents has been obtained, i.e., it is equivalent to a never examined set and its size is suboptimal). Instead, part of the original window is left unexamined at the end of the epoch, to be included as part of the next window. Should this happen during the ith epoch, then a comparison of the initial windows $(\tau_i,\ \tau_i + w_i]$ and $(\tau_{i+1},\ \tau_{i+1} + w_{i+1}]$, which were enabled at the beginnings of two successive epochs, would reveal an overlap, i.e., $\tau_{i+1} < \tau_i + w_i$. In this case, we say that the right-hand end of the ith window has regressed during the epoch.

This type of regression is problematic for our queueing theoretic methodology because it induces a correlation between the service times and interarrival times in the queueing system we use to obtain waiting time statistics. Thus, eliminating this correlation by introducing an *augmented* (by the amount of the window regression) epoch length as the service time in the queueing system is probably the most important contribution of this paper. This augmentation allows us to construct an equivalent system (with respect to waiting time statistics) where customer arrival points are regularly spaced within a busy period. It is interesting to note that the Poisson traffic assumption—which motivated the window regression rule

[17] Expressions for the conditional means and standard deviations of the epoch length and the final delay are provided in [29].

[18] With the SWA, the only approximation we make in the analysis of the FCFS 0.487 Algorithm is in using discrete- rather than continuous-time models for the queueing systems from which we obtain the statistics of the lag. The service time for these queues is the *augmented* epoch length (as we will see in Section IV), which, in addition to the length of the epoch generated, includes time equal to the size of the part of the window that has been left unexamined, and therefore, it is not necessarily an integral number of slots. An alternate approach would be to obtain numerical bounds to the solution to this queueing system using a "relaxation" method, such as [7].

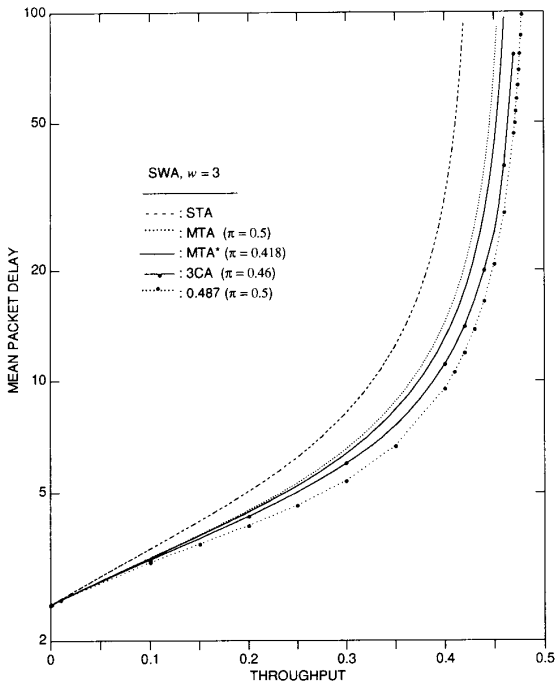

Fig. 6. Throughput–delay curves for the STA, the MTA, the optimally biased MTA (MTA*), the 3CA, and the FCFS 0.487 Algorithm.

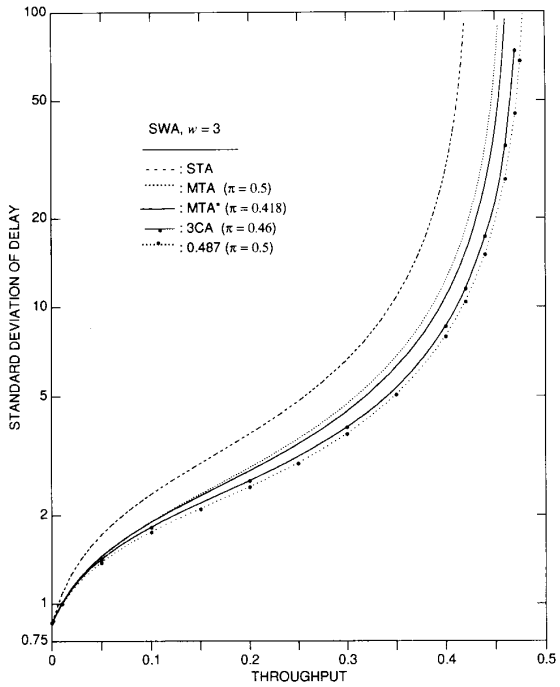

Fig. 7. Standard deviation of the delay for the STA, the MTA, the optimally biased MTA (MTA*), the 3CA, and the FCFS 0.487 Algorithm.

in the FCFS 0.487 Algorithm in the first place—also helps us solve the queueing systems we construct via our augmentation technique since it guarantees the independence of the service times.

In addition, we have to deal now with arrival-time addressing (to obtain a true FCFS system) which requires joint treatment of the initial and final components of the delay, and the nonsimplified Window Algorithm, to avoid the unnecessary "rest" periods.

A. Epoch Length

The standard approach in the literature for obtaining epoch length statistics for the FCFS 0.487 Algorithm has been to condition on the multiplicity of the initial conflict, as we did in Section III for the 3CA. However, here we follow our functional equation (FE) approach, i.e., we construct a FE for the PGF of epoch lengths, conditioned on the enabled set being Poisson distributed with parameter $x \triangleq \lambda w$. We consider it appropriate to show (at least) the initial steps in the construction. We actually consider the extension of the 0.487 Algorithm in the case of static[19] biased splitting, i.e., if the interval $(\tau, \tau + w]$ of the arrival time axis is considered for splitting, the two subsets will be $(\tau, \tau + \pi w]$ and $(\tau + \pi w, \tau + w]$, of length πw and $\bar{\pi} w = (1 - \pi)w$, respectively. The analysis of this variation comes essentially at no "cost" in derivation effort, and actually clarifies the method. Also, the extension leads to slightly higher capacity than the algorithm in [6].

We denote by $Q(w, z)$ the PGF of epoch lengths,[20] and the conditional PGF's for epochs starting with a conflict and nonempty epochs by $Q^{(c)}(w, z)$ and $Q^{(\bar{i})}(w, z)$, respectively. Considering the two subsets in a possible splitting, the set of all possible events is $\{i, s, c\} \times \{i, s, c\}$, where i denotes no busy points in the set (idle), s denotes a single busy point (success), and c denotes two or more busy points (conflict). By inspection, we obtain the following recursion:[21]

$$\begin{aligned} Q(w, z) &= \Pr\{(i,i) \text{ or } (i,s) \text{ or } (s,i)\} z \\ &\quad + \Pr\{(i,c)\} z Q^{(c)}(\bar{\pi} w, z) \\ &\quad + \Pr\{(s,s) \text{ or } (s,c)\} z^2 Q^{(\bar{i})}(\bar{\pi} w, z) \\ &\quad + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\} z Q^{(c)}(\pi w, z) \\ &= F(x) z + \Phi(\pi x) \bar{F}(\bar{\pi} x) z Q^{(c)}(\bar{\pi} w, z) \\ &\quad + \Psi(\pi x) \bar{\Phi}(\bar{\pi} x) z^2 Q^{(\bar{i})}(\bar{\pi} w, z) \\ &\quad + \bar{F}(\pi x) z Q^{(c)}(\pi w, z). \end{aligned}$$

[19] Unlike the dynamic optimization described in [21].

[20] Because we will need w as a parameter separate from λ, we use w instead of x as the first argument of $Q(\cdot, z)$ and the other transforms used in this section, with λ implied. Note also that we use w as both a "free" variable denoting the current size of an interval under examination (window), and as the size of the (maximum) initial window which is constant (and subject to optimization).

[21] Recall that, in the case of a conflict in a left subset, the FCFS 0.487 Algorithm leaves the right subset unexamined (to be part of the next interval).

We can obtain unconditional PGF's by using the identities

$$\bar{F}(x)Q^{(c)}(w,z) = Q(w,z) - zF(x) \quad \text{and}$$

$$\bar{\Phi}(x)Q^{(\bar{i})}(w,z) = Q(w,z) - z\Phi(x).$$

After substitutions, we obtain the FE sought:

$$\begin{aligned} Q(w,z) = {} & zQ(\pi w, z) \\ & + [z\Phi(\pi x) + z^2\Psi(\pi x)]Q(\bar{\pi} w, z) \\ & + (z - z^2)F(x) \\ & + (z - z^3)\Psi(\pi x)\Phi(\bar{\pi} x) \\ & - z^2 F(\pi x). \end{aligned} \tag{8}$$

This FE can be solved by substituting a power series expansion for $Q(w,z)$ and equating coefficients. Furthermore, by differentiating this FE with respect to z and setting $z = 1$, we obtain a FE for the mean epoch length:

$$\begin{aligned} L(w) = {} & 1 + L(\pi w) + F(\pi x)L(\bar{\pi} w) - F(x) - \Phi(\pi x) \\ & - \Psi(\pi x)\Phi(\bar{\pi} x) \end{aligned} \tag{9}$$

which can again be solved with the power series method.[22] Similarly, we can obtain the following FE for the second moment of the epoch length $H(w)$:

$$\begin{aligned} H(w) = {} & 1 + H(\pi w) + F(\pi x)H(\bar{\pi} w) + 2L(\pi w) \\ & + 2[\Phi(\pi x) + 2\Psi(\pi x)]L(\bar{\pi} w) \\ & - 3F(x) - 3\Phi(\pi x) - 5\Psi(\pi x)\Phi(\bar{\pi} x). \end{aligned} \tag{10}$$

Notice, however, that we are not yet ready to obtain the statistics of the (unconditional) epoch length in the case of the WA because we need the proportions of epochs generated by windows of various sizes. We will be able to compute these proportions after the development of a queueing model for the system in Section IV-C.

B. Capacity

The capacity of a CRA is the same with either the WA or the SWA, since for throughputs close to the capacity, all windows are "full size." Therefore, we can concentrate on the SWA for this investigation. We will use the standard renewal argument to obtain the throughput of the algorithm as the ratio of the mean number of successful transmissions per epoch to the mean epoch length. We have just computed the latter above. For the former, we have to essentially repeat the work to obtain FE's on the PGF and the expectation of packets transmitted during an epoch, denoted by $M(x,z)$ and $N(x)$, respectively (always, assuming an initial interval with busy points Poisson distributed with parameter x).[23] Following the above definitions, we have

$$\begin{aligned} M(x,z) = {} & \Pr\{(i,i)\}z^0 + \Pr\{(i,s) \text{ or } (s,i)\}z \\ & + \Pr\{(s,s)\}z^2 + \Pr\{(i,c)\}M^{(c)}(\bar{\pi} x, z) \\ & + \Pr\{(s,c)\}zM^{(c)}(\bar{\pi} x, z) \\ & + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\}M^{(c)}(\pi x, z) \end{aligned}$$

which, after some manipulations, takes the form

$$\begin{aligned} M(x,z) = {} & M(\pi x, z) \\ & + [\Phi(\pi x) + \Psi(\pi x)][M(\bar{\pi} x, z) - 1]. \end{aligned}$$

Differentiating, we get the FE for the mean:

$$N(x) = N(\pi x) + F(\pi x)N(\bar{\pi} x). \tag{11}$$

The throughput of the protocol can then be expressed as the ratio of the expected useful work per epoch to the expected length of the epoch, i.e., as $N(x)/L(x)$. Maximizing this ratio with respect to x for $\pi = 0.5$, i.e., the original algorithm with symmetric splitting, we obtain as capacity $\lambda^* \approx 0.4871$ at $x^* \approx 1.266$. Note that $x^* = \lambda^* w^*$, and thus the optimal window size is $w^* \approx 2.6$. However, we can also optimize over π. In that case, $\lambda^* \approx 0.48757$ at $x^* \approx 1.271$ and $\pi^* = 0.475$. In [21], the current window size is optimized at every step, through dynamic programming, which improves the capacity to $\lambda^* \approx 0.48776$. This last variation of the FCFS 0.487 Algorithm, with capacity rounded to three decimal places of 0.488, is the most efficient multiple-access protocol.[24]

C. Delay Analysis

We are now going to present a model of the protocol as a queueing system, which not only will provide the analysis for the statistics of the lag, i.e., the main component of the packet delay, but also the statistics for the relative proportions of epochs generated (and packets transmitted) from "large" and "small" window sizes in the case of the WA.[25] In Fig. 8, we define the following three components of the packet delay, the initial delay t_0, the lag t_1, and the delay in the epoch of transmission (or final delay) t_2. Notice that now, because of the overlap of the initial windows, we must be careful with packets that are part of more than one initial window; they must be included in the window that leads to their successful transmission; t_0 is then the time until the right end of the initial size of that window. For example, for the packet shown in Fig. 8, even though its arrival falls within the initial size of the ith window, it is really part of the $(i+1)$st window.

[22] Actually, one can numerically solve FE's of this form quite easily, by recursive application of the equation, stopping when the argument has become small enough to use the boundary condition, in this case, $L(0) = 1$.

[23] Recall that the algorithm, after a conflict in a left subset, "drops" the right subinterval, and therefore not all packets of the initial interval get transmitted successfully. Thus, the method for calculating the capacity that we have used in Section II does not directly apply. To apply that method, we need to substitute the mean *augmented* epoch length instead of the epoch length in (1).

[24] But check [35] for additional schemes and commentary.

[25] Except for certain cases, such as $w = 2$, this is an approximation because a small proportion of the windows will have an intermediate size —see Fig. 9.

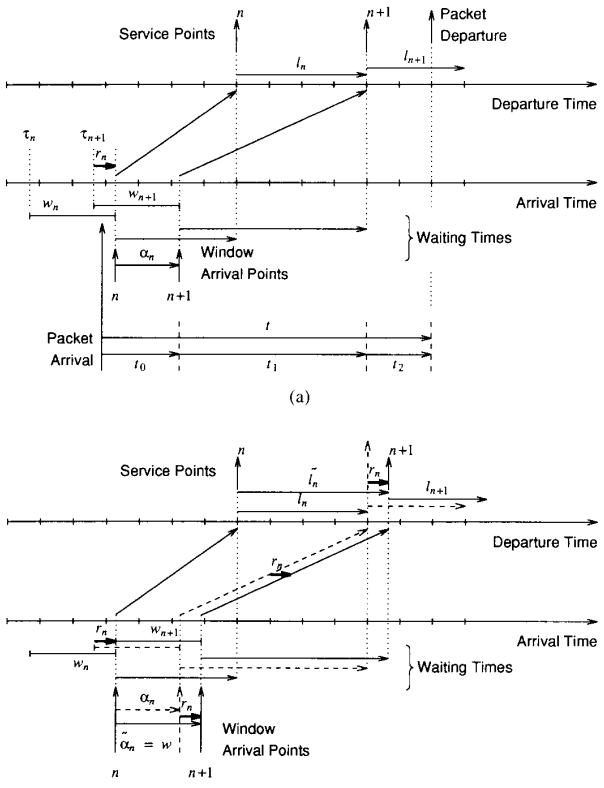

Fig. 8. The transformation used to model the lag as the waiting time in a $D/G/1$ queue. (In addition, the definitions of the components of the delay are illustrated.) (a) Original queueing system (interarrival times $\alpha_n \leq w$). (b) Transformed system (interarrival times $\tilde{\alpha}_n = w$). Dashed lines show the corresponding metrics in the original system.

Otherwise, the definitions of the components of the delay are identical to those used for algorithms that resolve the full window, i.e., as presented in Section II.

1) The Transformation for the Lag: The first step in obtaining the statistics of the lag is to view it as the waiting time in a queueing system where windows are the customers and their conflict resolution epoch lengths are their service times. If there were no regression on the arrival time axis, and the windows were all of constant size (i.e., the SWA were used), then this would have led to a $D/G/1$ queue, as in the case of [25]. Let us concentrate on the SWA for now to simplify the description of the transformation.

a) Simplified Window Access—The Unenhanced FCFS 0.487 Algorithm: Recall that in our discussion of window algorithms, we defined the nth window to be the interval $(\tau_n, \tau_n + w_n]$. Up to this point, we have assumed that our separability condition $\tau_{n+1} = \tau_n + w_n$ holds. We now address the *window regression* feature in the 0.487 Algorithm that makes it nonseparable according to our definition. That is, in accordance with Fig. 8, we now assume that $\tau_{n+1} = \tau_n + w_n - r_n$, where r_n represents the portion of the nth window that is left unexamined at the end of the epoch. We note that the distribution of r_n is nonnegative, and that it is *positively correlated* with the epoch length l_n. This complicates the approach of modeling t_1 as the waiting time in a queueing system since we are faced with a queueing problem which has both an *irregular* sequence of interarrival times $\alpha_n = w_n - r_n$, and *dependence* between one customer's service time l_n and the next customer's interarrival time α_{n+1}. These complications can be avoided by transforming our system into an *augmented* queueing problem, in which we use $\tilde{l}_n \triangleq l_n + r_n$ as the augmented service time, and $\tilde{\alpha}_{n+1} \triangleq \alpha_{n+1} + r_n$ as the augmented interarrival time. Substituting these augmented quantities into (2), we see that

$$\begin{aligned}\omega_{n+1} &= \max\left\{\omega_n + \tilde{l}_n - \tilde{\alpha}_{n+1}, 0\right\} \\ &= \max\{\omega_n + (l_n + r_n) - (\alpha_{n+1} + r_n), 0\} \\ &= \max\{\omega_n + l_n - \alpha_{n+1}, 0\},\end{aligned}$$

so it is clear that this transformation does not introduce any approximations into our analysis. Furthermore, since $\tilde{\alpha}_n \equiv w$, it is clear that the transformation has decoupled the customer interarrival process from the service time process. The "cost" of this change is that we must now solve for the joint statistics of l_i and r_i. In the Appendix, we obtain the following FE for the transform of the augmented epoch length:[26]

$$\begin{aligned}\tilde{Q}(w, z) &= z^{1+\overline{\pi}w}\tilde{Q}(\pi w, z) \\ &\quad + [z\Phi(\pi x) + z^2\Psi(\pi x)]\tilde{Q}(\overline{\pi}w, z) \\ &\quad + (z - z^2)F(x) + (z - z^3)\Psi(\pi x)\Phi(\overline{\pi}x) \\ &\quad - z^{2+\overline{\pi}w}F(\pi x). \qquad (12)\end{aligned}$$

b) Window Access—The Enhanced FCFS 0.487 Algorithm: Just as we saw in Section II-B2) for the STA, an alternate approach to finding the waiting time statistics is to apply the "Moving Server" transformation, in which we consider only conflict windows as the customers in an $M/G/1$ queue. And, just as we saw in Section II-C for the STA, the "Moving Server" approach can be extended to accommodate the Window Algorithm, which leads us to consider an $M/G/1$ queue with "generalized" busy periods. As before, the technique is exact only under some very specific conditions, but has proven to be extremely accurate in almost all cases.

To use this approach, we need to specify the statistics of the window that initiates the Generalized Busy Period (GBP). This is always going to be a one slot long window. On the other hand, all windows of the GBP, except possibly the last, are of the maximum size w. The only intermediate window sizes can appear at the end of the GBP, as the GBP is completing, but the lag has not grown to w yet. The windows in this last category (and the packets that are transmitted during their epochs) are a very small proportion of all windows, as can be seen from the area between the two curves in Fig. 9. A basic approx-

[26] Note that even though we present $Q(w, z)$ as a PGF, it is really a Laplace transform because the r_i's are not integers. A more detailed discussion is provided in the Appendix.

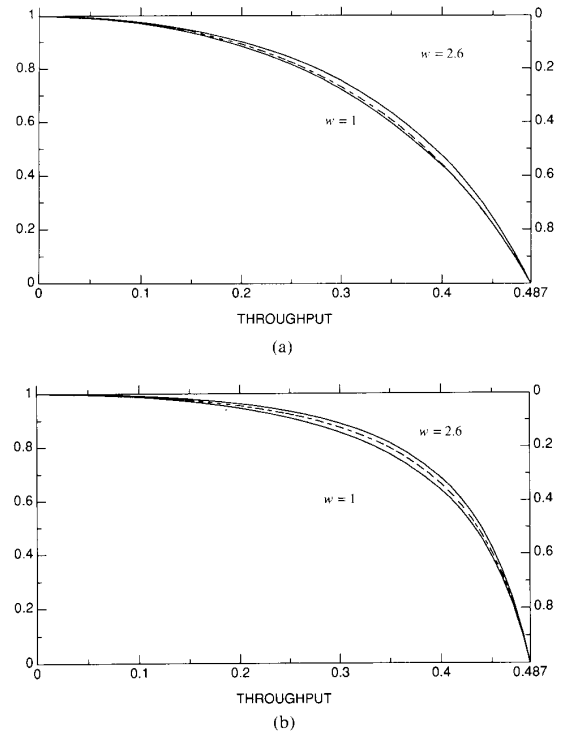

Fig. 9. (a) Simulation results for the proportion of epochs generated from windows of various sizes (FCFS 0.487 Algorithm, WA). The dashed line shows the analytic approximate dichotomy between "small" and "large" windows. (b) Simulation results for the proportion of packets transmitted during epochs generated from windows of various sizes (FCFS 0.487 Algorithm, WA). The dashed line shows the analytic approximate dichotomy between "small" and "large" windows.

imation we make in the analysis is to assume that windows are either of size 1 or w. The result of this extremal window approximation, as obtained from the analysis that follows, is the dashed line shown in Fig. 9, introducing a dichotomy for the window size. The final result for the transform of the lag is

$$W(w,z) = \frac{w - \tilde{L}(w)}{\tilde{L}(1) - 1} \frac{\tilde{Q}(1,z) - z}{z^w - \tilde{Q}(w,z)}. \tag{13}$$

The mean $T_1(w)$ and the second moment of the lag $T_1^{(2)}(w)$ can be obtained by differentiating $W(w,z)$. They can also be obtained by referring to the metrics of the GBP discrete-time $M/G/1$ queue. For example, the mean can be expressed as

$$T_1(w) = \frac{1}{2}\left[b_0(1 + C_0^2) - 1\right] + \frac{\rho}{2(1-\rho)}\left[b(1 + C^2) - 1\right]$$

where b_0 and b are the means and C_0^2 and C^2 are the squared coefficients of variation of the service time of the first customer of the GBP and a typical customer, respectively.

2) The Statistics of the Sum of the Initial and Final Delay: There is an additional complication in the analysis of the FCFS 0.487 Algorithm because of its use of arrival time addressing (which makes it FCFS in the first place). The problem is that under this addressing scheme, the initial packet delay in the window of arrival t_0 and the final packet delay, i.e., the delay of the packet in the epoch of transmission t_2, are correlated, and thus we cannot obtain distribution or higher moment results by treating them independently and then multiplying the PGF's or the Laplace transforms.

In order to obtain the exact statistics of the packet delay $t \triangleq t_0 + t_1 + t_2$, we first obtain the statistics of $t_{0,2} \triangleq t_0 + t_2$; then, because t_1 is independent of $t_{0,2}$, the statistics of the total delay can be obtained by multiplying the transforms. We denote[27] by $J^*(w,s)$ the Laplace transform of $t_{0,2}$ assuming a window of size w and traffic intensity λ. In a way very similar to the derivations of FE's for the PGF's of the epoch length and the augmented epoch length, we obtain, in the Appendix, the following FE:

$$N(x)J^*(w,s) = \Psi(x)(z - z^2)U^*(w,s) + [\Phi(\pi x)z + \Psi(\pi x)z^2]N(\bar{\pi}x)J^* \cdot(\bar{\pi}w, s) + z^{1+\bar{\pi}w}N(\pi x)J^*(\pi w, s) \tag{14}$$

where $z = e^{-s}$ and $U^*(w,s)$ is the transform of a uniform distribution in $[0,w)$. Notice that a solution for the product $N(x)J^*(w,s)$ is obtained which, combined with the solution for $N(x)$ from (11), provides the Laplace transform of $t_{0,2}$. By differentiation, we obtain the following FE's for the first and second moments of the final delay,[28] $T_{0,2} \triangleq E[t_{0,2} \mid w]$ and $T_{0,2}^{(2)}(w) \triangleq E[t_{0,2}^2 \mid w]$, respectively:

$$T_{0,2}(w) = 1 + T_{0,2}(w/2) + \frac{w/2 + \Psi(x/2)}{1 + F(x/2)} - \frac{\Psi(x)}{N(x)} \tag{15}$$

$$T_{0,2}^{(2)}(w) = 1 + T_{0,2}^{(2)}(w/2) + 2\left[1 + \frac{w/2 + \Psi(x/2)}{1 + F(x/2)}\right]T_{0,2}(w/2) + \frac{w^2/4 + w + 3\Psi(x/2)}{1 + F(x/2)} - (3 + w)\frac{\Psi(x)}{N(x)}. \tag{16}$$

3) The Total Delay: To obtain statistics for the system that are not conditioned on the size of the window, we need the proportion of epochs generated from "large"

[27] We decided to not make the dependence on λ explicit in order to simplify the notation.

[28] To simplify the expressions, we consider the unbiased splitting case, where $\pi = \bar{\pi} = 0.5$.

windows u given by

$$u = \frac{\frac{b_0}{1-\rho}}{\frac{1}{p_0} + \frac{b_0}{1-\rho}} = \frac{\rho_0}{1-\rho+\rho_0} \tag{17}$$

where ρ and ρ_0 are given by (4), but substituting $\tilde{L}(\ )$ for $L(\ )$. Notice that $1/p_0$ is the mean idle period and $b_0/(1-\rho)$ is the mean (generalized) busy period, both measured in windows. The proportion of packets transmitted in epochs generated from "large" windows ν is obtained by weighing the above proportion with the mean number of packets successfully transmitted from a window of the respective size, i.e., $N(w\lambda)$ for a window of size w:

$$\nu = \frac{\rho_0 N(w\lambda)}{(1-\rho)N(\lambda) + \rho_0 N(w\lambda)}. \tag{18}$$

The Laplace transform of the total packet delay can then be obtained as

$$D^*(w,s) = \nu W(w, e^{-s}) J^*(w,s) + \bar{\nu} J^*(1,s) \tag{19}$$

and the mean as

$$T(w) = \nu[T_{0,2}(w) + T_1(w)] + \bar{\nu} T_{0,2}(1). \tag{20}$$

D. Results and Comparisons

Given that our solution involves approximations in the case of the 0.487 Algorithm, we have also performed an extensive simulation study to determine the accuracy of our methodology.[29] The results are particularly encouraging. In Fig. 9, we show with solid lines the proportions of epochs generated [Fig. 9(a)] and packets transmitted [Fig. 9(b)] from windows of the minimum and maximum size (i.e., $w = 1$ and $w = 2.6$, respectively) as a function of throughput. The area between the two solid lines represents windows of intermediate size. Dashed lines show the split induced by our extremal window approximation, i.e., our assumption that only windows of sizes $w = 1$ and 2.6 are used. Both the small size of the area between the solid lines and its closeness to the predicted dichotomy according to our extremal window approximation (especially viewed on a packet basis) support the use of this approximation.

In Fig. 10, we plot the mean and the standard deviation of the epoch length, and compare them to simulation results (the small horizontal lines show 95% confidence intervals). The agreement is extremely good. Notice that these are unconditional measures, and that the extremal approximation for the window sizes has been incorporated into the analytical results. An interesting observation is that the standard deviation of the epoch length is consistently lower than the mean, even for throughputs close to the capacity of the algorithm.

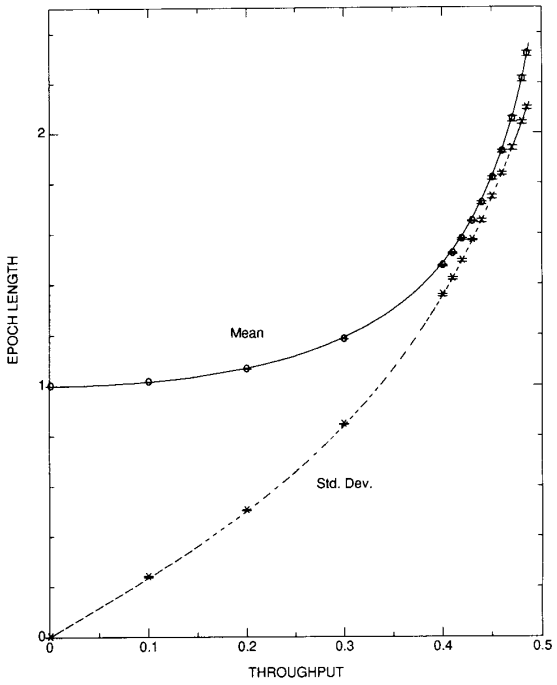

Fig. 10. Comparison of analytical with simulation results for the mean and standard deviation of the epoch length for the FCFS 0.487 Algorithm ($w = 2.6$). The effects of the "extremal window size" approximation are almost unnoticeable for the epoch length statistics.

We compare analytical with simulation results for the mean and the standard deviation of the packet delay (and its components $t_{0,2}$ and t_1) in Figs. 11 and 12, respectively. Our analysis for $t_{0,2}$ is exact, apart from the weighing of the conditional results (i.e., we use the extremal window size approximation instead of the full window size distribution). The simulation results shown in this figure completely support our analytical results (and the extremal window size approximation). More interestingly, and importantly, we obtain similarly good results for the mean lag. Notice that in this case, more approximations are involved. The extremal window size approximation does not only specify the weighing now, but also characterizes the queueing system; furthermore, the solution for the queueing system is only approximate because the augmented epoch length we use as service time for our queueing systems (which is not an integer, in general) does not satisfy the conditions we need for the "Moving Server" technique to produce exact results. However, the accuracy of the approximation, as shown in this figure, can be judged as excellent. Therefore, it is not surprising that we obtain excellent results for the total mean packet delay.

Numerical results and relative error estimates are provided in Table I for the mean and in Table II for the standard deviation of the packet delay. From these, we

[29] In [10], the Part-and-Try Algorithm is analyzed, but with binary rather than ternary feedback. Thus, the results obtained there are not directly comparable with the results we obtained here. This delay distribution methodology has also been presented and applied to other systems in [8] and [9].

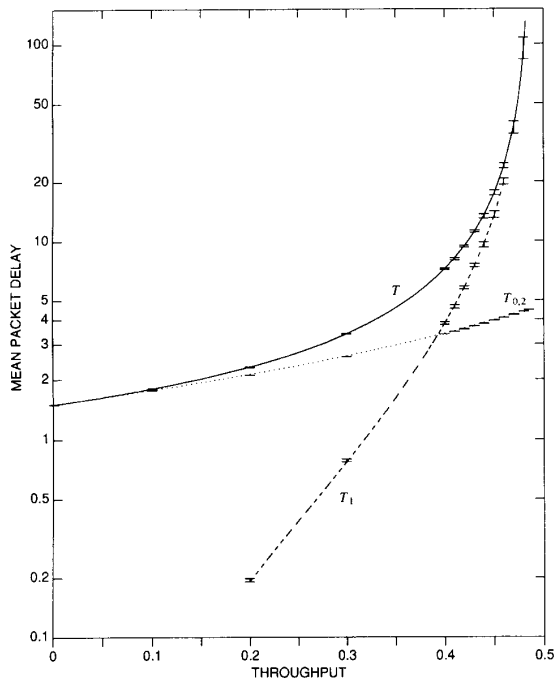

Fig. 11. Comparison of analytical with simulation results (95% confidence intervals) for the total mean packet delay T and its components $T_{0.2}$ and T_1 for the FCFS 0.487 Algorithm (WA, $w = 2.6$). (The T_1 curve has been limited for clarity.)

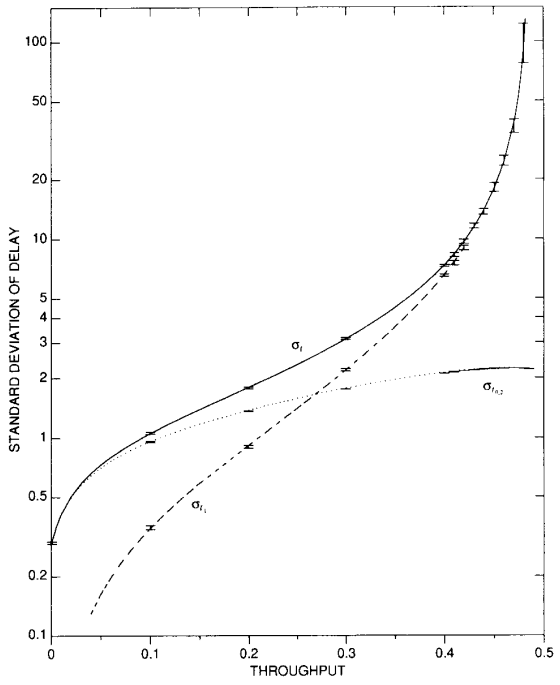

Fig. 12. Comparison of analytical with simulation results for the standard deviation of the packet delay (and its components) for the FCFS 0.487 Algorithm (WA, $w = 2.6$). (The σ_{t_1} curve has been limited for clarity.)

see that the maximum error is less than 1.3% for the mean and less than 6.6% for the standard deviation, and both appear at very high loads (for throughputs of 0.48 and above), which suggests that the simulation estimate might be suspect. Furthermore, the errors are in both directions, suggesting that they are probably due to random variations in the simulation estimates, rather than systematic analytic bias.

In Table I and Fig. 13, we compare our results with bounds on the mean delay for the FCFS 0.487 Algorithm obtained by Tsybakov and Likhanov [34], by Huang and Berger [13], and by Georgiadis *et al.* [7]. Our results agree very well with those of Georgiadis *et al.*, with the difference between our estimate and their midpoint less than 0.5% apart throughout the throughput range (Table I), and also with the upper bound of Tsybakov and Likhanov (Fig. 13). For large values of throughput, the results of Huang and Berger are inaccurate.

V. Conclusions

We have briefly presented our queueing theoretic methodology for analysis of random access protocols by first concentrating on the "separable" class. "Separable" protocols consist of a conflict resolution algorithm (CRA) in combination with a window algorithm for selecting the initial set of contenders at the beginning of each conflict resolution epoch. A key characteristic of protocols in this class is that the CRA does not terminate before the entire window has been examined. This is a less general class of protocols than can be treated using the regenerative method of Georgiadis *et al.* [7], [10]. However, what is interesting about this methodology is that, when it applies, the information obtained from the performance analysis is essentially complete since we get the (steady-state) *distribution* of the packet delay exactly, from which, of course, all other statistics can be derived. This permits us to see the effects of minor differences between protocols, such as random versus arrival-time addressing, or the use of biased and/or d-ary splitting, and also to easily incorporate changes in the environment due to channel errors, variable message lengths, carrier sensing, and collision detection in a unified way.

There are, however, other window algorithms, notably the FCFS 0.487 Algorithm, which do not satisfy the conditions for separability. Due to the importance of the 0.487 Algorithm, we have followed two different directions in order to analyze its delay performance. We first introduce an approximate algorithm, the Three-Cell Algorithm (3CA), for which we are able to obtain exact results (under many circumstances) and which provides pessimistic bounds for the performance of the 0.487 Algorithm. In particular, we obtain the distribution of the packet delay, and thus moments and percentiles. Our results are exact for the SWA (with any window size, but free of any numerical component for $w = 2$), and also for the WA with $w = 2$. Although $w = 2$ is not the optimal

TABLE I
COMPARISON OF RESULTS FOR THE MEAN DELAY FOR THE FCFS 0.487 ALGORITHM (WA)

	Analysis					Simulation 95% Confidence Interval		Comparisons % Difference Between Our Method and...	
	Our	Georgiadis *et al.* [7]		Huang and Berger [13]					
λ	Method (Approx.)	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Mean	Dev. (+/−)	[7] (Midpoint)	Simulation (Midpoint)
0.01	1.525	1.5253	1.5255					0.000	
0.05	1.637	1.6348	1.6388					0.000	
0.10	1.805	1.796	1.8130	1.8	1.8	1.807	0.004	−0.055	0.111
0.20	2.311	2.270	2.352	2.29	2.3	2.312	0.005	0.000	0.043
0.25	2.732	2.66	2.80					−0.073	
0.30	3.395	3.270	3.525	3.33	3.43	3.384	0.026	0.059	−0.325
0.35	4.581	4.358	4.8151					0.131	
0.40	7.224	6.779	7.670	6.53	7.31	7.191	0.160	0.000	−0.459
0.42	9.433	8.801	10.11	7.91	9.3	9.440	0.223	0.243	0.074
0.44	13.57	12.584	14.63	9.94	12.6	13.46	0.35	0.272	−0.817
0.45	17.34	16.030	18.754					0.299	
0.46	23.92	22.043	25.95	14.00	19.2	23.78	1.03	0.317	−0.597
0.48	92.91	85.086	101.452	22.4	38.5	94.09	13.11	0.385	1.252
0.487	5691.	5200.	6228.					0.403	

TABLE II
COMPARISON OF RESULTS FOR THE STANDARD DEVIATION OF THE PACKET DELAY FOR THE FCFS 0.487 ALGORITHM

	Analysis		Simulation 95% Confidence Interval		
		% Difference from		Deviation (+/−)	
λ	Our Method (Approx.)	Simulation Midpoint	Midpoint Value	Absolute	% of Midpoint
0.001	0.302782	2.64	0.295	0.004	1.36
0.1	1.056721	−0.31	1.060	0.011	1.04
0.2	1.796372	0.81	1.782	0.016	0.90
0.3	3.134142	0.00	3.134	0.030	0.96
0.4	7.343612	0.64	7.297	0.097	1.33
0.41	8.352253	1.01	8.269	0.181	2.19
0.42	9.657946	0.01	9.657	0.233	2.41
0.43	11.416115	−1.35	11.572	0.320	2.77
0.44	13.914207	1.76	13.674	0.451	3.30
0.45	17.749131	−2.20	18.149	0.907	5.00
0.46	24.398088	−1.51	24.771	1.426	5.76
0.47	38.789444	4.53	37.108	2.836	7.64
0.48	93.548870	−6.52	100.071	22.702	22.69

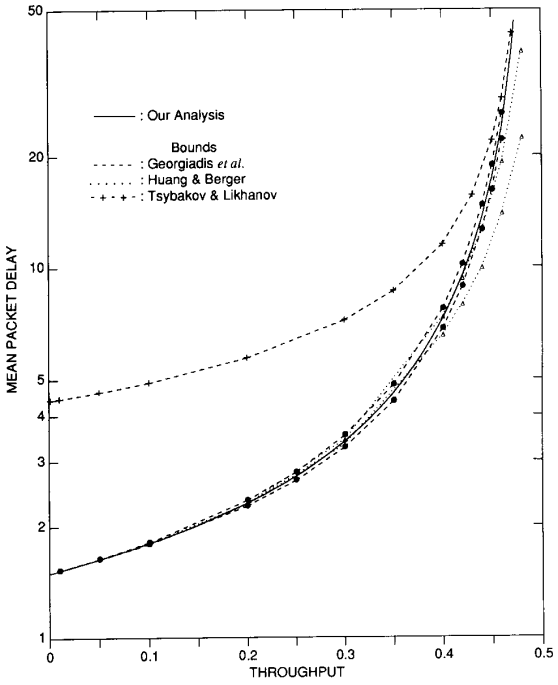

Fig. 13. Comparison of our analytical results for the mean packet delay for the FCFS 0.487 Algorithm ($w = 2.6$) with the bounds provided by Tsybakov and Likhanov in [34], Huang and Berger in [13], and Georgiadis *et al.* in [7].

window size in terms of capacity, it is nevertheless a reasonable choice and, indeed, the one suggested by Gallager in [6].

Then, we consider the "full" FCFS 0.487 Algorithm (including the minor extension to static biased splitting, which raises the capacity to 0.48757), with both simplified and nonsimplified window access (i.e., both the unenhanced and enhanced versions of the 0.487 Algorithm using the terminology of [13]). Our main contribution here is the introduction of the *augmented* epoch length transformation, i.e., we add to the epoch length the part of the window that the 0.487 Algorithm "drops" in order to counteract the regression on the arrival time axis that would result. Then, in the case of the SWA, the lag of the algorithm is equivalent to the waiting time in a $D/G/1$ system which has windows (of the initial size, typically $w = 2.6$) as customers, interarrival time the (initial) window size, and service time the augmented epoch length. This transformation is exact. However, the $D/G/1$ queue is not a discrete-time system anymore (unlike that for the 3CA) because the augmentation of the epoch length is done by various fractions of the window size. (If we were in a position to obtain the solution for this queueing system exactly, our analysis for the unenhanced 0.487 Algorithm would be exact.)

In the case of the nonsimplified Window Algorithm, the situation is further complicated by the existence of various intermediate window sizes between the minimum ($w = 1$) and the maximum (typically, $w = 2.6$). Since, however, the proportion of intermediate window sizes is very small, we have introduced the extremal window approximation,

which allows us to model the lag as the waiting time in a "Moving Server" discrete time $M/G/1$ queue. This last step involves an additional approximation since service times for this last queue (i.e., the augmented epoch lengths for conflict windows) are not exact multiples of $w-1$, as is required. However, the accuracy of our results is excellent for both the mean and variance of the packet delay. In combination with our results for the unenhanced 0.487 Algorithm and for the 3CA (which provide upper bounds for the delay), they provide a comprehensive system of analytic results.

Appendix
Functional Equations for the FCFS 0.487 Algorithm

We derive here a functional equation (FE) on the transform of the augmented epoch length $\tilde{l}$, i.e., the length of an epoch generated by a window of (initial) size w, augmented by the length of the part of the window that remains unexamined at the end of the epoch, if any. Notice that the augmented epoch length is not necessarily an integer anymore, and thus a Laplace transform is the appropriate transform:

$$\tilde{Q}^*(w,s) \triangleq \int_{-\infty}^{+\infty} e^{-st} \frac{d}{dt} \Pr\{\tilde{l} \le t\}\, dt.$$

However, it is convenient to define an intermediate form $\tilde{Q}(w,z)$, very similar to a PGF, so that

$$\tilde{Q}^*(w,s) = \tilde{Q}(w, e^{-s}).$$

We follow the approach we used in Section IV-A to obtain a FE for the PGF of epoch lengths, conditioned on the enabled set being Poisson distributed with parameter $x \triangleq \lambda w$. Again, considering the two subsets at a candidate level of splitting, the set of all possible events is $\{i,s,c\} \times \{i,s,c\}$, where i denotes no busy points in the set (idle), s denotes a single busy point (success), and c denotes two or more busy points (conflict). By inspection, we obtain the following relation:

$$\begin{aligned}
\tilde{Q}(w,z) &= \Pr\{(i,i) \text{ or } (i,s) \text{ or } (s,i)\} z \\
&\quad + \Pr\{(i,c)\} z \tilde{Q}^{(c)}(\bar{\pi} w, z) \\
&\quad + \Pr\{(s,s) \text{ or } (s,c)\} z^2 \tilde{Q}^{(\tilde{i})}(\bar{\pi} w, z) \\
&\quad + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\} z \tilde{Q}^{(c)}(\pi w, z) z^{\bar{\pi} w} \\
&= F(x) z + \Phi(\pi x) \bar{F}(\bar{\pi} x) z \tilde{Q}^{(c)}(\bar{\pi} w, z) \\
&\quad + \Psi(\pi x) \bar{\Phi}(\bar{\pi} x) z^2 \tilde{Q}^{(\tilde{i})}(\bar{\pi} w, z) \\
&\quad + \bar{F}(\pi x) \tilde{Q}^{(c)}(\pi w, z) z^{1+\bar{\pi} w}.
\end{aligned}$$

We can obtain expressions for the unconditional PGF's by using

$$\bar{F}(x) \tilde{Q}^{(c)}(w,z) = \tilde{Q}(w,z) - z F(x) \quad \text{and}$$

$$\bar{\Phi}(x) \tilde{Q}^{(\tilde{i})}(w,z) = \tilde{Q}(w,z) - z \Phi(x).$$

After substitutions, we obtain the FE sought:

$$\begin{aligned}
\tilde{Q}(w,z) &= z^{1+\bar{\pi} w} \tilde{Q}(\pi w, z) \\
&\quad + [z\Phi(\pi x) + z^2 \Psi(\pi x)] \tilde{Q}(\bar{\pi} w, z) \\
&\quad + (z - z^2) F(x) + (z^2 - z^3) \Psi(\pi x) \Phi(\bar{\pi} x) \\
&\quad - z^{2+\bar{\pi} w} F(\pi x).
\end{aligned}$$

Differentiating this FE with respect to z and setting $z = 1$, we obtain the following FE for the mean augmented epoch length:

$$\begin{aligned}
\tilde{L}(w) &= (1 + \bar{\pi} w)[1 - F(\pi x)] + \tilde{L}(\pi w) \\
&\quad + F(\pi x) \tilde{L}(\bar{\pi} w) - F(x) \\
&\quad - \Psi(\pi x)[1 - \Phi(\bar{\pi} x)].
\end{aligned}$$

Similarly, we can obtain the following FE for the second and third moments of the augmented epoch length $\tilde{H}(w)$ and $\tilde{V}(w)$, which are required for the computation of the mean and variance of the packet delay. For simplicity, we present the unbiased splitting case only:

$$\begin{aligned}
\tilde{H}(w) &= (1 + w/2)^2 + [1 + F(w/2)] \tilde{H}(w/2) \\
&\quad + [2 + w + 2\Phi(w/2) + 4\Psi(w/2)] \tilde{L}(w/2) \\
&\quad - 3F(w) - 3\Phi(w/2) - 5\Psi(w/2)\Phi(w/2) \\
&\quad - (w^2/4 + 2w) F(w/2)
\end{aligned}$$

$$\begin{aligned}
\tilde{V}(w) &= (1 + w/2)^3 + [1 + F(w/2)] \tilde{V}(w/2) \\
&\quad + 3\left[1 + \frac{w}{2} + F(w/2) + \Psi(w/2)\right] \tilde{H}(w/2) \\
&\quad + 3\left[(1+w)^2 + \Phi(w/2) + 4\Psi(w/2)\right] \tilde{L}(w/2) \\
&\quad - 7F(w) - 7\Phi(w/2) - 19\Psi(w/2)\Phi(w/2) \\
&\quad - (w^3/8 + 3w^2/4 + 6w) F(w/2).
\end{aligned}$$

We obtain a FE on the product $N(x)J^*(w,s)$ by considering an exhaustive set of possible events at a candidate level of splitting, exactly as we obtained the FE for the PGF of the epoch lengths $Q(x,z)$. For convenience, we use the z variable as the transform of one unit of time instead of e^{-s}. Since we concentrate on a "tagged" packet, there is always at least one transmission in the sets considered here.

$$
\begin{aligned}
N(x)J^*(w,s) &= \Pr\{(i,s) \text{ or } (s,i)\} zU^*(w,s) \\
&\quad + \Pr\{(i,c)\} zN^{(c)}(\bar{\pi}x)J^{*(c)}(\bar{\pi}w,s) \\
&\quad + \Pr\{(s,s)\}[z^{2+\bar{\pi}w}U^*(\pi w,s) + z^3U^*(\bar{\pi}w,s)] \\
&\quad + \Pr\{(s,c)\}[z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + z^2N^{(c)}(\bar{\pi}x)J^{*(c)}(\bar{\pi}w,s)] \\
&\quad + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\} z^{1+\bar{\pi}w}N^{(c)}(\pi x) \\
&\quad \cdot J^{*(c)}(\pi x, z) \\
&= \Psi(x)zU^*(w,s) \\
&\quad + \Phi(\pi x)\bar{F}(\bar{\pi}x)zN^{(c)}(\bar{\pi}x)J^{*(c)}(\bar{\pi}w,s) \\
&\quad + \Psi(\pi x)\Psi(\bar{\pi}x)[z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + z^3U^*(\bar{\pi}w,s)] \\
&\quad + \Psi(\pi x)\bar{F}(\bar{\pi}x)[z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + z^2N^{(c)}(\bar{\pi}x)J^{*(c)}(\bar{\pi}w,s)] \\
&\quad + \bar{F}(\pi x)z^{1+\bar{\pi}w}N^{(c)}(\pi x)J^{*(c)}(\pi x,z) \\
&= \Psi(x)zU^*(w,s) \\
&\quad + \Phi(\pi x)z[N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad - \Psi(\bar{\pi}x)zU^*(\bar{\pi}w,s)] \\
&\quad + \Psi(\pi x)\Psi(\bar{\pi}x)[z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + z^3U^*(\bar{\pi}w,s)] \\
&\quad + \Psi(\pi x)\bar{F}(\bar{\pi}x)[z^{2+\bar{\pi}w}U^*(\pi w,s)] \\
&\quad + \Psi(\pi x)z^2[N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad - \Psi(\bar{\pi}x)zU^*(\bar{\pi}w,s)] \\
&\quad + z^{1+\bar{\pi}w}[N(\pi x)J^*(\pi w,s) \\
&\quad - \Psi(\pi x)zU^*(\pi w,s)] \\
&= \Psi(x)zU^*(w,s) \\
&\quad + \Phi(\pi x)z[N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad - \Psi(\bar{\pi}x)zU^*(\bar{\pi}w,s)] \\
&\quad - \Psi(\pi x)\Phi(\bar{\pi}x)z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + \Psi(\pi x)z^2N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad + z^{1+\bar{\pi}w}N(\pi x)J^*(\pi w,s) \\
&= \Psi(x)zU^*(w,s) \\
&\quad + \Phi(\pi x)zN(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad + \Psi(\pi x)z^2N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad - \Phi(\pi x)\Psi(\bar{\pi}x)z^2U^*(\bar{\pi}w,s) \\
&\quad - \Psi(\pi x)\Phi(\bar{\pi}x)z^{2+\bar{\pi}w}U^*(\pi w,s) \\
&\quad + z^{1+\bar{\pi}w}N(\pi x)J^*(\pi w,s) \\
&= \Psi(x)zU^*(w,s) - z^2[\Phi(\pi x)U^*(\bar{\pi}w,s) \\
&\quad + \Psi(\pi x)\Phi(\bar{\pi}x)z^{\bar{\pi}w}U^*(\pi w,s)] \\
&\quad + [\Phi(\pi x)z + \Psi(\pi x)z^2]N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad + z^{1+\bar{\pi}w}N(\pi x)J^*(\pi w,s) \\
&= \Psi(x)(z-z^2)U^*(w,s) \\
&\quad + [\Phi(\pi x)z + \Psi(\pi x)z^2]N(\bar{\pi}x)J^*(\bar{\pi}w,s) \\
&\quad + z^{1+\bar{\pi}w}N(\pi x)J^*(\pi w,s)
\end{aligned}
$$

where we have used the identity

$$
\begin{aligned}
\Psi(x)U^*(w,s) &= \Phi(\pi x)\Psi(\bar{\pi}x)U^*(\bar{\pi}w,s) \\
&\quad + \Psi(\pi x)\Phi(\bar{\pi}x)z^{\bar{\pi}w}U^*(\pi w,s).
\end{aligned}
$$

REFERENCES

[1] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[2] J. I. Capetanakis, "Generalized TDMA: The multiaccessing tree protocol," *IEEE Trans. Commun.*, vol. COM-27, pp. 1476–1484, Oct. 1979.

[3] ____, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 505–515, Sept. 1979.

[4] J. W. Cohen, *The Single Server Queue*. Amsterdam: North-Holland, 1969.

[5] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet, "Analysis of a stack algorithm for random multiple-access communication," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 244–254, Mar. 1985.

[6] R. G. Gallager, "Conflict resolution in random access broadcast networks," in *Proc. AFOSR Workshop Commun. Theory Appl.*, Sept. 1978, pp. 74–76.

[7] L. Georgiadis, L. F. Merakos, and P. Papantoni-Kazakos, "A method for the delay analysis of random multiple-access algorithms whose delay process is regenerative," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1051–1062, July 1987.

[8] L. Georgiadis and M. Paterakis, "Delay distribution analysis of window random-access algorithms," in *Proc. 26th Conf. Decision Contr.*, Los Angeles, CA, Dec. 1987, pp. 703–707.

[9] ____, "Performance analysis of window type random-access algorithms for packet radio networks, in *Proc. IEEE INFOCOM '89*, Ottawa, Canada, Apr. 1989, pp. 505–511.

[10] ____, "Bounds on the delay distribution of window random-access algorithms," *IEEE Trans. Commun.*, 1993.

[11] J. F. Hayes, "An adaptive technique for local distribution," *IEEE Trans. Commun.*, vol. COM-26, pp. 1178–1186, Aug. 1978.

[12] J. Huang and T. Berger, "Delay analysis of interval-searching contention resolution algorithms," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 264–273, Mar. 1985.

[13] ____, "Delay analysis of 0.487 contention resolution algorithms," *IEEE Trans. Commun.*, vol. COM-34, pp. 916–926, Sept. 1986.

[14] J. Hunter, *Mathematical Techniques of Applied Probability Vol. 2, Discrete Time Models: Techniques and Applications*. New York: Academic, 1983.

[15] L. Kleinrock, *Queueing Systems, Vol. I, Theory*. New York: Wiley-Interscience, 1975.

[16] J. L. Massey, "Collision-resolution algorithms and random-access communications," in *Multi-User Communications*, G. Longo, Ed. New York: Springer-Verlag, 1981.

[17] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," *IEEE Trans. Inform. Theory*, vol. IT-31, Mar. 1985.

[18] L. Merakos and C. Bisdikian, "Delay analysis of the n-ary stack random-access algorithm," *IEEE Trans. Inform. Theory*, vol. 34, pp. 931–942, Sept. 1988.

[19] M. L. Molle and L. Kleinrock, "Virtual time CSMA: Why two clocks are better than one," *IEEE Trans. Commun.*, vol. COM-33, Sept. 1985.

[20] M. L. Molle, "Analysis of a class of distributed queues with applications," *Performance Eval.*, vol. 9, pp. 271–286, Aug. 1989.

[21] J. Mosely and P. A. Humblet, "A class of efficient contention resolution algorithms for multiple access," *IEEE Trans. Commun.*, vol. COM-33, pp. 145–151, Feb. 1985.

[22] A. G. Pakes, "Some conditions for ergodicity and recurrence of Markov chains," *Oper. Res.*, vol. 17, pp. 1058–1061, 1969.

[23] M. Paterakis and P. Papantoni-Kazakos, "A simple window random access algorithm with advantageous properties," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1124–1130, Sept. 1989.

[24] G. C. Polyzos, M. L. Molle, and A. N. Venetsanopoulos, "Delay analysis of tree conflict resolution algorithms: The non-homoge-

neous case," in *Proc. IEEE GLOBECOM* '85, New Orleans, LA, Dec. 1985, pp. 1504–1509.

[25] ____, "Performance analysis of finite nonhomogeneous population tree conflict resolution algorithms using constant size window access," *IEEE Trans. Commun.*, vol. COM-35, pp. 1124–1138, Nov. 1987.

[26] G. C. Polyzos and M. L. Molle, "A generalized busy period approach to the delay analysis of window access tree conflict resolution algorithms," in *Proc. IEEE ICC* '87, Seattle, WA, June 1987.

[27] ____, "Delay analysis of a window tree conflict resolution algorithm in a local area network environment," in *Proc. ACM SIGMETRICS* '87, Banff, Alta., Canada, May 1987.

[28] G. C. Polyzos, "A queueing theoretic approach to the delay analysis for a class of conflict resolution algorithms," Tech. Rep. CSRI-224, Comput. Syst. Res. Inst., Univ. Toronto, Toronto, Ont., Canada, Mar. 1989.

[29] G. C. Polyzos and M. L. Molle, "Delay analysis for the FCFS 0.487 conflict resolution algorithm," Tech. Rep. CS 90-179, Dep. Comput. Sci. Eng., Univer. California at San Diego, LaJolla, Oct. 1990.

[30] L. D. Servi, "$D/G/1$ queues with vacations," *Oper. Res.*, vol. 34, pp. 619–629, July–Aug. 1986.

[31] *IEEE Trans. Inform. Theory*, Special Issue on Random-Access Communications, vol. IT-31, Mar. 1985.

[32] B. S. Tsybakov and V. A. Mikhailov, "Free synchronous packet access in a broadcast channel with feedback," *Problemy Peredachi Informatsii*, vol. 14, pp. 32–59, Oct.–Dec. 1978.

[33] ____, "Random multiple packet access: Part-and-try algorithm," *Problemy Peredachi Informatsii*, vol. 16, no. 4, pp. 65–79, Oct.–Dec. 1980.

[34] B. S. Tsybakov and N. B. Likhanov, "Upper bound for the delay in a multiple-random-access system with a splitting algorithm," *Problemy Peredachi Informatsii*, vol. 18, pp. 76–84, Oct.–Dec. 1982.

[35] B. S. Tsybakov, "Survey of USSR contributions to random multiple-access communications," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 143–165, Mar. 1985.

[36] B. S. Tsybakov and N. B. Likhanov, "Some new random multiple-access algorithms," *Problemy Peredachi Informatsii*, vol. 21, pp. 69–89, Apr.–June 1985. (Also in *Proc. Int. Symp. Inform. Theory*, Brighton, England, June 1985, p. 102.)

[37] J. K. Wolf, "Born again group testing: Multiaccess communications," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 185–191, Mar. 1985.