Conflict Multiplicity Estimation and Batch Resolution Algorithms

ISRAEL CIDON, MEMBER, IEEE, AND MOSHE SIDI, SENIOR MEMBER, IEEE

Abstract—**The standard model of a multiple access channel with ternary feedback is considered. When packets of a batch of k nodes initially collide, it is assumed that no *a priori* statistical information about k is available. A new algorithm is presented and analyzed that enables the nodes to compute a statistical estimate of k. Combining the estimation procedure with tree algorithms leads to batch resolution algorithms that resolve conflicts more efficiently than any other reported to date. Both complete resolution and partial resolution algorithms are presented.**

I. Introduction

We consider the following standard model of a multiple access channel. A large number of geographically dispersed nodes communicate through a common channel. Any node can generate and transmit data on the channel. Transmissions start at integer multiples of the unit of time and last one unit of time, also called a "slot." When k nodes transmit simultaneously, the success of the transmission depends on k:

- if $k = 0$, then no transmissions were attempted;
- if $k = 1$, then the transmission succeeds;
- if $k \geq 2$, there is a *conflict*, meaning that the transmissions interfere destructively so that none succeeds.

Prior to the next slot, all nodes receive the feedback $0, 1, 2+$, indicating whether the *multiplicity* of the conflict is $k = 0$, $k = 1$, or $k \geq 2$, respectively. This is known as the ternary feedback model.

Algorithms for successful transmissions of packets under this model have been the subject of numerous papers ([1]–[10] and many others). The standard assumption used in these studies is that the total arrival process of new packets into the system forms a stationary Poisson process. Under this assumption, conflict resolution algorithms that yield a stable system for arrival rates under 0.462 and 0.487 packets per slot were suggested in [4] and [1], [2], [5], respectively.

In this paper we take another point of view which does not assume knowledge of the statistical characteristics of the generation process of new packets. Specifically, when k is the number of nodes whose packets initially collide, we assume that no *a priori* statistical information about k is available. The set of k nodes whose packets initially collide is called a *batch*. We are interested in developing algorithms that efficiently resolve conflicts among the nodes of the batch.

A. Batch Resolution Algorithms and Their Efficiency

A *batch resolution algorithm* is a distributed algorithm performed by the nodes of the batch to successfully transmit (resolve) the packets of the initial conflict (batch). Batch resolution algorithms should be *independent* of k as no *a priori* statistical knowledge on the actual batch size k is known. Two types of batch resolution algorithms are distinguishable:

1) complete resolution—all packets of the batch are successfully transmitted,
2) partial resolution—only a fraction of the packets of the batch are successfully transmitted.

The batch resolution algorithm is executed only *once*. It starts with an initial conflict of $k \geq 2$ nodes (the batch) and ends at some globally known point in time. The time elapsed from the point when the algorithm starts until the point it ends is called the *batch resolution interval* (BRI), which is measured in slots. At the end of the BRI each node in the network is aware that the BRI ended, and each node of the batch knows whether its packet was successfully transmitted during this BRI or not.

An efficient batch resolution algorithm should resolve a large number of packets from the batch using a short BRI for each conflict multiplicity k. To compare different batch resolution algorithms, it is convenient to use a single asymptotic measure termed *efficiency*. Let L_k be the average length of a BRI given that it starts with an initial conflict of multiplicity k, and let M_k be the average number of packets successfully transmitted during this BRI (for complete resolution $M_k = k$). The *efficiency* of a batch resolution algorithm is defined as $E = \liminf_{k \to \infty} \{M_k / L_k\}$. We wish to devise batch resolution algorithms that are highly efficient.

Efficiency, as defined earlier, has a crucial significance when nodes are accessing the shared channel in the obvious manner described in [4]. According to the obvious access algorithm, batch resolution algorithms are sequen-

Manuscript received August 25, 1986; revised March 15, 1987. This work was supported in part by the Bat-Sheva de Rothshild fund. This paper was presented in part at INFOCOM '86, Miami, FL, April 1986.

I. Cidon is with the Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

M. Sidi was with the Thomas J. Watson Research Center, Yorktown Heights, NY, on leave from the Electrical Engineering Department, Technion, Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 8719313.

0018-9448/88/0100-0101$01.00 ©1988 IEEE

tially performed. After a previous BRI is ended, a new batch, consisting of all packets that arrived to the system during that BRI and all packets of the previous batch that have not been successfully transmitted, is resolved. In Section VI we show that, under weak conditions on the arrival process, when the average arrival rate of new packets is less than the efficiency, the system is stable and, therefore, the average rate at which packets are successfully transmitted is equal to the average arrival rate.

Nonobvious access algorithms have been presented in [1]–[4]. These algorithms can behave poorly when applied to arrival streams with different statistics than those for which they were tailored. This has been demonstrated in [8] for a bursty arrival process. The reason is that the parameters of the nonobvious access algorithms [1]–[4] are finely tuned to the Poisson arrival process. (We note here that the dynamic tree algorithm of [3] is also included in this category.)

The first batch resolution algorithm (though described and analyzed with the Poisson assumption on the arrival process) was presented in [3]; it has efficiency of 0.346. This algorithm was later modified in [4] to yield an efficiency of 0.375, and further optimized in [7] to yield an efficiency of 0.381. A better batch resolution algorithm with an approximated efficiency of 0.430 has been introduced by Greenberg and Ladner [8]. In a later paper [10], independently of our work, the algorithm of [8] has been improved to obtain an efficiency as high as 0.468. All the algorithms are complete resolution algorithms.

B. Objectives and Outline of Paper

Our main objective in this paper is to present two new batch resolution algorithms. The first, presented in Section IV, is a complete resolution algorithm and has an efficiency as high as 0.468. The second, presented in Section V, is a partial resolution algorithm (that can be transformed into a complete resolution algorithm with the same efficiency, as explained in Section V-C) and has an efficiency as high as 0.487.

Our algorithms are inspired by the hybrid algorithm first introduced in [8]. They consist of two basic phases, an *estimation phase* devoted to estimating the conflict multiplicity of the batch, and a *resolution phase* when the conflict is resolved. The idea of estimating the conflict multiplicity has been introduced by Greenberg and Ladner in [8]. They suggested a rapid distributed estimation procedure which enables calculation of an estimate k^* to k and uses $O(\log_2(k))$ slots on the average to do that when the batch is of size k. This estimate yielded an algorithm with efficiency of about 0.430. To increase the efficiency, the estimation procedure of [8] has been slowed down [10] to $O(\log_a(k))$ slots on the average where a is a parameter of the algorithm ($1 < a \leq 2$), and it has been shown in [10] that when $a \to 1$, the efficiency of the resulting algorithm approaches 0.468. Asymptotic analysis of the first two moments of the estimate k^* has been presented in [10].

An important contribution in this paper is the introduction in Section II of a simpler estimation procedure completely different from that in [8], [10]. For large k it is slower than the earlier procedure and uses $O(k)$ slots on the average. Yet, even with our relatively slow estimation procedure, only a negligible fraction of the BRI is devoted to this phase. The simplicity of our estimation procedure allows us to provide a complete characterization of the probability distribution of k^* (rather than the asymptotic analysis of its moments). Another advantage of our estimate is that it is tighter in the sense that its variance is of $O(k)$ (rather than $O(k^2)$ in [8], [10]), which enables us to use the law of large numbers to prove its normalized asymptotic accuracy and to increase the efficiency of the overall resolution algorithm. Further comparison between our estimation scheme and the estimation scheme of [8], [10] is provided in Section VII.

The motivation for estimating the conflict multiplicity is that the estimate k^* can be utilized to help in efficient resolution of the conflict between the nodes of the batch during the resolution phase. Algorithms similar to those presented in [1], [4] are applied, and since the estimate k^* is quite accurate, it is possible to make this phase highly efficient and thereby to compensate for the relatively slow estimation phase. In addition, we address the issue of further tuning of our algorithms so that they will also perform well when k is not too large (see Section IV-C).

The most important property of the algorithms presented here is their *robustness*, namely, that their parameters provide a high stability region for many arrival processes. In Section VI we give some examples of specific arrival processes and show that the efficiency of an algorithm determines the stability region of the system.

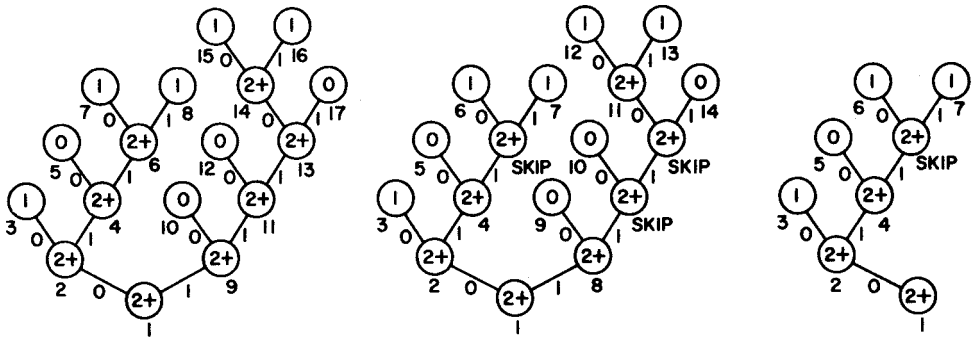
II. Binary Tree Algorithms—A Review

We start with a brief review of binary tree algorithms for conflict resolution whose understanding will be helpful when we will describe our batch resolution algorithms.

A. Basic Binary Tree Algorithm

We first describe the basic binary tree algorithm due to Capetanakis [3] in a recursive fashion, as was done in [4]. Suppose that k nodes transmit simultaneously to the channel. If the resulting feedback indicates that k is 0 or 1, then there is no conflict (conflict is resolved). Otherwise, each of the conflicting nodes tosses a (possibly biased) binary coin, at which point nodes whose coins turned up 0 retransmit to the channel. This can result in a conflict that these nodes now resolve. Next, nodes whose coins turned up 1 transmit to the channel. A conflict can result that these nodes now resolve. An example of a typical binary tree that describes the evolution of the algorithm is depicted in Fig. 1(a). The slot numbers are written under the circles in which the events appear.

Massey [4] introduced a simple way to implement this algorithm with only two local counters per node. It has been shown that an unbiased coin is optimal in this algorithm that yields efficiency as high as 0.346.



(a) (b) (c)

Fig. 1. Binary tree algorithms. (a) Basic. (b) Modified. (c) Clipped.

B. Modified Binary Tree Algorithm

Massey [4] and Tsybakov and Mikhailov [2] found a simple way to improve the basic algorithm—by avoiding sure collisions. The modified algorithm is identical to the basic algorithm except that whenever nodes can infer at some slot that a conflict must occur in the next slot, they "skip" that slot by tossing the coin before the conflict occurs. A typical example for such a situation is when a slot with a conflict is followed by an empty slot, in which case it is clear that the next slot will contain a conflict (see Fig. 1(b)).

Let L_n be the average number of slots required to resolve a conflict among n nodes with the modified binary tree algorithm. Let b be the probability that a coin tossed at a node will turn up 0, and let $Q_i(n) = \binom{n}{i} b^i (1-b)^{n-i}$ be the probability that i of n nodes will turn up 0. Then it is not difficult to see that L_n can be computed recursively via [4]:

$$L_0 = L_1 = 1 \tag{1a}$$

$$L_n = 1 - Q_0(n) + \sum_{i=0}^{n} Q_i(n)(L_i + L_{n-i}), \qquad n \geq 2. \tag{1b}$$

When $b = 0.4175$ (the best bias as shown in [7]), we see from (1) that $L_2 = 4.4143$ and $L_3 = 6.8847$. As in [4] it can be shown that

$$L_n \leq C_u n - 1, \qquad n \geq 4 \tag{2}$$

where $C_u = 2.623$. We see from (2) that the efficiency of the modified algorithm with a biased coin ($b = 0.4175$) is 0.381 [7].

Before proceeding, we should mention that both the basic and the modified binary tree algorithms are complete batch resolution algorithms. When they start with a conflict of n nodes, all n nodes transmit their packets successfully during execution. In the examples of Fig. 1(a) and (b), we see that $n = 5$.

C. Clipped Binary Tree Algorithm

The clipped binary tree algorithm has been independently introduced by several authors [1], [2], [5]. It is identical to the modified binary tree algorithm except that it is stopped (the tree is clipped) whenever two consecutive successful transmissions follow a conflict (see Fig. 1(c)). This algorithm is a partial batch resolution algorithm since not necessarily all nodes of an initial batch transmit their packets successfully during its execution. Let L_n^G be the average number of slots required to perform the clipped binary tree algorithm given that it starts with an initial conflict of multiplicity n, and let M_n^G be the average number of packets successfully transmitted during its execution. Let $Q_i(n)$ be as defined in the previous subsection. Then L_n^G and M_n^G are recursively computed via [5]:

$$L_0^G = L_1^G = 1 \tag{3a}$$

$$L_n^G = 1 + Q_0(n) L_n^G + Q_1(n)\left(1 + L_{n-1}^G\right) + \sum_{i=2}^{n} Q_i(n) L_i^G, \qquad n \geq 2 \tag{3b}$$

$$M_0^G = 0; \; M_1^G = 1 \tag{3c}$$

$$M_n^G = Q_0(n) M_n^G + Q_1(n)\left(1 + M_{n-1}^G\right) + \sum_{i=2}^{n} Q_i(n) M_i^G, \qquad n \geq 2. \tag{3d}$$

It is also very easy to see that $L_n^G \leq L_n$.

The clipped binary tree algorithm was originally designed to deal with Poisson arrival process. It is stable for arrival rates up to 0.4871 when $b = 0.5$. A slightly modified version of this algorithm is stable for arrival rates up to 0.4877 [9]. However, its performance as a batch resolution algorithm is poor. Its efficiency approaches to 0 when the batch size increases.

III. The Estimation Procedure

Suppose that an initial conflict of multiplicity $k \geq 2$ occurs, and recall that no *a priori* statistical knowledge about k is available. We wish to devise a distributed estimation procedure that generates a random function K^* of k whose value k^* gives appropriate indication of the value of k. Clearly, the procedure should not use too many slots. The strategy that we adopt is to resolve a small portion of the batch and to accumulate the number of successful transmissions resulted. To that end, we let each

of the k colliding nodes transmit to the channel with probability $p > 0$. Thus the k colliding nodes are partitioned into two sets E and D, where E consists of those that transmitted and D the rest. Clearly, $|E|+|D|=k$ where $|A|$ is the cardinality of set A. If the resulting slot is empty or contains a successful transmission, we conclude that $|E|=0$ or $|E|=1$, respectively. If a conflict occurs, it is known that $|E|\geq 2$, and then the nodes in E use one of the complete batch resolution algorithms presented in Sections II-A and II-B to resolve the conflict between the nodes in E. At the end of this part of the estimation phase we know the exact value of $|E|$ by accumulating the number of successful transmissions during the resolution. This value is called j. Then k^* is computed via $k^* = j/p$.

Remarks: 1) Note that during the estimation procedure, all nodes in E transmit their packets successfully. Consequently, only packets of D should be further resolved.

2) For simplicity and to attain an unbiased estimate, we let $k^*=0$ when $j=0$. In the resolution phase the fact that $k\geq 2$ will be taken into account.

Let J be an integer-valued random variable that expresses the number of nodes in E. Given the batch size k, J is binomially distributed with parameter p. Therefore, we have the following.

Lemma 1: We have

1) $P(J=j|k)=\binom{k}{j}p^j(1-p)^{k-j}, \qquad 0\leq j\leq k$
2) $E[J|k]=kp$
3) $\text{var}(J|k)=kp(1-p)$.

From Lemma 1 and by using Tchebyceff's inequality we obtain the following.

Lemma 2: For any $\epsilon>0$ we have

$$P\left(\left|\frac{J}{k}-p\right|\geq\epsilon|k\right)\leq\frac{p(1-p)}{\epsilon^2 k}.$$

Let K^* be a real-valued random variable that expresses our estimate upon k. The following theorem states the tightness of our estimate K^* of k.

Theorem 1: The following hold:

1) $P(K^*=k^*|k)=\binom{k}{j}p^j(1-p)^{k-j}, \qquad 0\leq j\leq k,$ $k^*=j/p$
2) $E[K^*|k]=k$
3) $P\left(\left|\frac{K^*}{k}-1\right|\geq\epsilon|k\right)\leq\frac{1-p}{\epsilon^2 k}.$

The proof follows directly from Lemmas 1 and 2.

As we see from Theorem 1, for sufficiently large batch size ($k\to\infty$), our estimate K^* of k is very tight. The next crucial issue regarding the estimation phase is the average number of slots required for its completion. This number depends on the specific batch resolution algorithm that is used by the estimation procedure. Again, the alternatives are the basic binary tree algorithm or the modified binary tree algorithm with unbiased or biased coin.

In this paper we assume that the modified binary tree algorithm (MBTA) with a biased coin ($b=0.4175$) is used.

Let L_j be the average number of slots required to complete the estimation procedure given that $|E|=j$. Then from (1) we see that

$$L_0=L_1=1 \tag{4a}$$

$$L_j=1-Q_0(j)+\sum_{i=0}^{j}Q_i(j)(L_i+L_{j-i}), \qquad j\geq 2 \tag{4b}$$

$$L_j\leq C_u\cdot j+1, \qquad j\geq 0 \tag{5}$$

where we recall that $C_u=2.623$.

Let $L^e(k)$ ($k\geq 2$) be the average number of slots required to complete the estimation procedure with a batch of k nodes. Then, we conclude the following from (5) and Lemma 1.

Theorem 2: The following holds:

$$L^e(k)\leq pC_u k+1.$$

From Theorem 2 we see that when $0<p\ll 1$, the average number of slots required to estimate k is quite small compared to k. It is also easy to realize that on the average, pk packets are successfully transmitted during the estimation phase.

IV. A Complete Resolution Algorithm

A. The Algorithm

In this section we present a new complete batch resolution algorithm. We exploit the observation made by Capetanakis [3] that binary tree algorithms are most efficient for conflicts of small multiplicity and adopt his idea of a *dynamic tree* to exploit the improved efficiency. To further improve the resolution phase, the MBTA of [4] is implemented.

By combining the estimation and the resolution procedure, we construct a hybrid batch resolution algorithm that consists of two phases. The first is devoted to estimating the batch size k according to our estimation procedure presented in Section III. Recall that at the beginning of this procedure the k colliding nodes are divided into two sets E and D, and that the nodes of E transmit their packets successfully during the estimation procedure. The second phase of the hybrid batch resolution algorithm is devoted to transmitting the packets of nodes in D successfully. To that end, all nodes in D pick a number in the range $1,2,\cdots,m$ uniformly and independently at random as soon as the estimation phase ends. Here m is a global parameter that is a function of the known number of nodes in E, i.e., $m=m(j)$, and, therefore, can be computed by all nodes. This divides the nodes of D into m distinct groups which are then individually and sequentially resolved using the MBTA. First, those that picked 1 are transmitted, and the resulting conflict is resolved using the MBTA. Second, those that picked 2 transmit, and so forth.

An appropriate choice of the function $m(j)$ to compute m is important in our hybrid algorithm. Evidently, our estimate on $|D|$ is k^*-j, and we choose to compute m as

a linear function of the estimate on $|D|$, i.e.,

$$m(j) = \max\{1, \lceil \alpha(k^* - j) - \beta \rceil\} = \max\{1, \lceil \gamma j - \beta \rceil\} \tag{6}$$

where $\gamma = \alpha(1-p)/p$ and $\lceil \cdot \rceil$ is the ceiling function. The parameter α will be specified later so that the resolution phase will be the most efficient for large k. The parameter β is significant only when k is not too large, and its effect will be discussed at the end of this section. Note that m is an integer-valued random variable whose probability distribution can be derived from that of j. However, the exact form of the probability distribution of m does not concern us here.

B. Analysis

We are now ready to compute the average length of the resolution phase $L_j^r(k)$ given a batch size k and given that $|E| = j$. It is clearly given by

$$L_j^r(k) = m(j) \sum_{i=0}^{k-j} \binom{k-j}{i} \left(\frac{1}{m(j)}\right)^i \cdot \left(1 - \frac{1}{m(j)}\right)^{k-j-i} L_i, \qquad m(j) \geq 1 \tag{7}$$

where the L_i are computed via (1) and the $m(j)$ via (6).

When $L_j^r(k)$ is averaged over j, we get $L^r(k)$—the average number of slots required for the resolution phase (and, therefore, to resolve all packets of the batch completely) given k. We prove the following theorem regarding $L^r(k)$.

Theorem 3: We have

$$\limsup_{k \to \infty} \{ L^r(k)/k \} \leq C_r$$

where $C_r = (1-p)(C_u - 0.488) = (1-p)2.135$.

Proof (sketch): In the Appendix we give a rigorous and formal proof of the theorem. Because of the many Epsilons needed in the rigorous proof, it is quite messy. Here we will sketch a less rigorous yet more insightful argument that will present the main idea of the proof very clearly.

Our point of departure is (7). From Lemmas 1 and 2 we conclude that for large k ($k \to \infty$), $j \simeq kp$ and $k^* \simeq k$ with very high probability. Therefore, $m \simeq \alpha(k-j)$ (β is negligible). Let $n = k - j \simeq k(1-p)$ and $D_i(n) = \binom{n}{i}(1/\alpha n)^i(1-(1/\alpha n))^{n-i}$. Then

$$L^r(k) = \alpha n \sum_{i=0}^{n} D_i(n) L_i. \tag{8}$$

In (8) we replaced approximate equality with very high probability by a strict equality. From (1) and (2) we recall that $L_0 = L_1 = 1$, $L_2 = 4.4143$, $L_3 = 6.8847$, and $L_i \leq C_u i - 1$, $i \geq 4$, where $C_u = 2.623$. Consequently,

$$\begin{aligned} L^r(k) &\leq \alpha n \sum_{i=0}^{n} D_i(n)(C_u i - 1) \\ &\quad + \alpha n \sum_{i=0}^{3} D_i(n)(L_i - C_u i + 1) \\ &= C_u n - \alpha n[1 - V(n)] \end{aligned} \tag{9}$$

where $V(n) = \sum_{i=0}^{3} D_i(n)\Delta_i$ and $\Delta_i = L_i - C_u i + 1$.

For large n (recall that as $k \to \infty$, $n \to \infty$), we obtain from (9)

$$V(n) \simeq e^{-1/\alpha} \sum_{i=0}^{3} \frac{\Delta_i}{i!\alpha^i}. \tag{10}$$

Therefore, we conclude from (8) that for large k,

$$L^r(k) \leq k(1-p)\left[C_u - \alpha\left(1 - e^{-1/\alpha} \sum_{i=0}^{3} \frac{\Delta_i}{i!\alpha^i}\right)\right]. \tag{11}$$

Using the values of Δ_i ($\Delta_0 = 2$, $\Delta_1 = -0.623$, $\Delta_2 = 0.1683$, $\Delta_3 = 0.0157$), we calculate the value of α that minimizes the right side of (9). This value is $\alpha = 0.786$, and we get

$$L^r(k) \leq k(1-p)(C_u - 0.488) \tag{12}$$

and therefore,

$$\limsup_{k \to \infty} \{ L^r(k)/k \} \leq C_r. \qquad \text{Q.E.D.}$$

Let L_k^1 be the total average number of slots required to resolve a batch of size k with the complete resolution algorithm of this section. Then $L_k^1 = 1 + L^e(k) + L^r(k)$ (the 1 counts the slot of the initial conflict of the batch), and therefore, from Theorems 2 and 3 we conclude the following.

Theorem 4: We have

$$E = \liminf_{k \to \infty} \{ k/L_k^1 \} \geq 1/[C_u - 0.488(1-p)].$$

Consequently, the efficiency of our complete resolution algorithm is as high as 0.468 (by 8.8 percent better than that in [8] and the same as that in [10]), and it becomes better as p becomes smaller. The result expressed in Theorem 4 shows that for large k the average number of slots required to resolve a conflict of a batch with k nodes is about $2.137k$. For comparison we recall that this number is $2.890k$, $2.623k$, and $2.342k$ for the basic tree algorithm, for the modified tree algorithm, and for the hybrid algorithm of [8], respectively.

Note that the optimal value of α that should be used is 0.786. This means that when the nodes of D are divided into m groups, each group would contain 1.27 packets on the average when the estimate on the number of nodes in D is accurate.

C. Effect of β

The parameter β has no effect on the efficiency of the algorithm when $k \to \infty$. However, it can be adjusted so that the algorithm will have good performance for small

values of k. This is done by keeping m small for small values of j. To show the effect of β, we plot the ratio $\{k/L_k^1\}$ versus k in Fig. 2. In that figure we used $\alpha = 0.786$ and $p = 0.1$ (from Theorem 4 we see that in this case $\liminf_{k\to\infty}\{k/L_k^1\} \geq 0.457$). From the figure we see that $\beta \simeq 8$ is a good choice when $p = 0.1$.

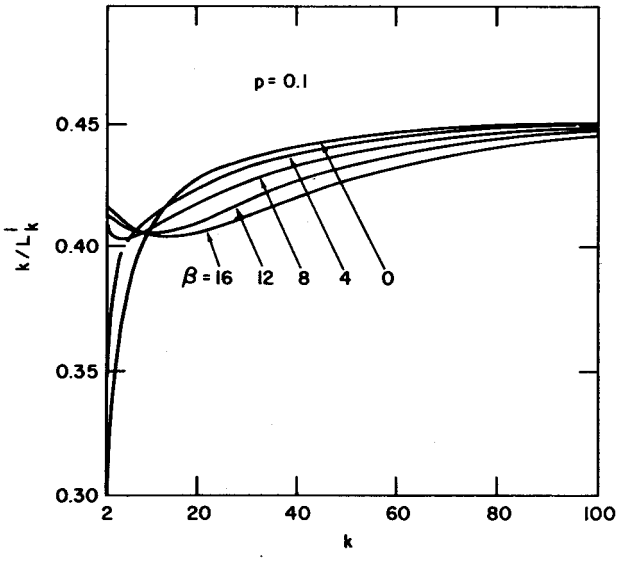


Fig. 2. Effect of β.

V. A Partial Resolution Algorithm

A. The Algorithm

In this section we present a partial batch resolution algorithm that is based on the clipped binary tree algorithm (CBTA). This algorithm is identical to the complete resolution algorithm presented in Section IV except that the nodes of D, after being divided into m distinct groups, perform the CBTA instead of the MBTA. This means that, first, those that picked 1 are transmitted and the resulting conflict is resolved using the CBTA. Second, those that picked 2 transmit, and so forth.

B. Analysis

We now compute the average length of the resolution phase $L_j^r(k)$ given a batch size k and given that $|E| = j$. As in (7) it is given by

$$L_j^r(k) = m(j)\sum_{i=0}^{k-j}\binom{k-j}{i}\left(\frac{1}{m(j)}\right)^i \cdot\left(1-\frac{1}{m(j)}\right)^{k-j-i} L_i^G, \qquad m(j)\geq 1 \quad (13)$$

where the L_i^G are computed via (3a) and (3b) and $m(j)$ via (6).

When $L_j^r(k)$ is averaged over j, we get $L^r(k)$—the average number of slots required for the resolution phase given k. We prove the following theorem regarding $L^r(k)$.

Theorem 5: Let $l_1 \geq 1$ be some finite integer. Let $\Delta_i^G = L_i^G - C_u i + 1$. Then

$$\limsup_{k\to\infty}\{L^r(k)/k\} \leq C_r(\alpha)$$

where

$$C_r(\alpha) \leq (1-p)\left[C_u - \alpha\left(1 - e^{-1/\alpha}\sum_{i=0}^{l_1}\frac{\Delta_i^G}{i!\alpha^i}\right)\right].$$

Proof: We give here only an informal proof. The formal proof is similar to the proof of Theorem 3. Similarly to (8), we have here $n \simeq k(1-p)$:

$$L^r(k) = \alpha n\sum_{i=0}^{n} D_i(n) L_i^G. \quad (14)$$

Therefore,

$$\begin{aligned} L^r(k) &= \alpha n\sum_{i=0}^{n} D_i(n)(C_u i - 1) \\ &\quad + \alpha n\sum_{i=0}^{n} D_i(n)\left(L_i^G - C_u i + 1\right) \\ &\leq C_u n - \alpha n\left[1 - V(n, l_1)\right] \end{aligned} \quad (15)$$

where $V(n, l_1) = \sum_{i=0}^{l_1} D_i(n)\Delta_i^G$.

In (15) we used the fact that $\Delta_i^G \leq 0$ for $i \geq 1$. We recall that for large n,

$$V(n, l_1) \simeq e^{-1/\alpha}\sum_{i=0}^{l_1}\frac{\Delta_i^G}{i!\alpha^i}, \quad (16)$$

and, therefore, for large k,

$$L^r(k) \leq k(1-p)\left[C_u - \alpha\left(1 - e^{-1/\alpha}\sum_{i=0}^{l_1}\frac{\Delta_i^G}{i!\alpha^i}\right)\right], \quad (17)$$

and Theorem 5 follows.

Recall that we are dealing with a partial resolution algorithm. As such, not all packets in D are successfully transmitted during the resolution phase. Let $M_j^r(k)$ be the average number of packets successfully transmitted during the resolution phase given a batch size k and given that $|E| = j$. Then

$$M_j^r(k) = m(j)\sum_{i=1}^{k-j}\binom{k-j}{i}\left(\frac{1}{m(j)}\right)^i \cdot\left(1-\frac{1}{m(j)}\right)^{k-j-i} M_i^G, \qquad m(j)\geq 1 \quad (18)$$

where the M_i^G are computed via (3c) and (3d) and $m(j)$ via (6).

When $M_j^r(k)$ is averaged over j, we get $M^r(k)$—the average number of packets successfully transmitted during the resolution phase given k. We prove the following theorem regarding $M^r(k)$.

Theorem 6: Let $l_2 \geq 1$ be some finite integer. Then

$$\liminf_{k\to\infty}\{M^r(k)/k\} \geq C_r'(\alpha)$$

where

$$C_r'(\alpha) = \alpha(1-p)e^{-1/\alpha}\sum_{i=1}^{l_2}\frac{M_i^G}{i!\alpha^i}.$$

Proof: We again briefly sketch here an informal proof. Similarly to (8), we have here

$$M^r(k) = \alpha n\sum_{i=1}^{n} D_i(n)M_i^G \tag{19}$$

where $n = k(1-p)$. Therefore,

$$M^r(k) \geq \alpha k(1-p)e^{-1/\alpha}\sum_{i=1}^{l_2}\frac{M_i^G}{i!\alpha^i}. \tag{20}$$

Q.E.D.

To complete the analysis of the partial batch resolution algorithm, we recall that $L^e(k) \leq pC_u k + 1$ and that during the estimation procedure an average of kp packets are successfully transmitted. Therefore, the total average number of slots required to execute the algorithm for a batch of size k is $L_k^2 = 1 + L^e(k) + L^r(k)$. The average number of packets successfully transmitted during its execution is $M_k^2 = kp + M^r(k)$. Consequently, we have the following.

Theorem 7: Let $C_r(\alpha)$ and $C_r'(\alpha)$ be computed as in Theorems 5 and 6, respectively. Then

$$E = \liminf_{k\to\infty}\{M_k^2/L_k^2\} \geq \frac{p + C_r'(\alpha)}{pC_u + C_r(\alpha)}.$$

The lower bound on the efficiency can be maximized by appropriately choosing α. We note that here the optimal value of α depends on the value of p (in the complete resolution algorithm the optimal α is independent of p). In Table I we summarize the values of the optimal α for various values of p along with the corresponding values of the lower bound. We also added a column indicating the fraction of packets successfully transmitted. To compute these numbers, we use $l_1 = l_2 = 15$.

TABLE I

p	Optimal α	Lower Bound	$\liminf_{k\to\infty}\{M_k/k\}$
$2\cdot10^{-1}$	0.807	0.457	0.928
10^{-1}	0.798	0.471	0.918
10^{-2}	0.791	0.485	0.908
10^{-3}	0.790	0.486	0.907
10^{-4}	0.790	0.487	0.907
$\to 0$	0.790	0.487	0.907

We conclude from the table that the efficiency of the partial resolution algorithm introduced in this section is as high as 0.487. Note also that about 91 percent of the packets in the batch are successfully transmitted during its execution.

C. From Partial to Complete Resolution Algorithm

Interesting, the partial resolution algorithm introduced and analyzed in this section can be modified into a complete resolution algorithm with efficiency higher than 0.468. In fact, it is possible to devise a complete resolution algorithm with efficiency arbitrarily close to 0.487.

The idea is the following. After applying the algorithm as described in Section V-A, there remain some nodes of D whose packets were not successfully transmitted. At this point we can apply the modified binary tree algorithm (described in Section II-B) on these remaining nodes, and as a consequence, all nodes of the initial batch will transmit their packets successfully. Thus the algorithm has been transformed into a complete resolution algorithm.

To calculate the efficiency of this complete resolution algorithm, let $P(i|k)$ be the probability that, given that the batch is of size k, i of the k nodes have not transmitted their packets successfully when the partial resolution algorithm has been executed. Then the total average number of slots required for the complete resolution algorithm just described is given by

$$L_k^3 = L_k^2 + \sum_{i=0}^{k} P(i|k)L_i \tag{21}$$

where the L_i are calculated via (1). Using (2), we have

$$L_k^3 \leq L_k^2 + \sum_{i=0}^{k} P(i|k)(C_u i + 1) = L_k^2 + \left(k - M_k^2\right)C_u + 1, \tag{22}$$

and simple calculation shows that the efficiency $E = \liminf_{k\to\infty}\{k/L_k^3\}$ of this algorithm is as high as 0.475.

If instead of applying the MBTA, the partial resolution algorithm is reexecuted on those nodes in D that have not transmitted their packets successfully, then only about one percent of the nodes of D would remain with packets. We may now apply the MBTA on these remaining nodes. The result is a complete resolution algorithm with efficiency as high as 0.485. Obviously, efficiencies arbitrarily close to 0.487 can be obtained. The foregoing result shows that for large k the average number of slots required to resolve a conflict of a batch with k nodes is about $2.054k$, faster than any other known algorithm.

VI. Stability Condition

To devise multiple access algorithms, one needs to specify the rules according to which nodes access the channel and the rules for resolving conflicts. In this section we assume that the obvious way to do that is used. A complete batch resolution algorithm is used to resolve conflicts. When a BRI ends, a new BRI with those packets that arrived during the previous BRI is started [4]. Let $L(i)$ $(i = 1, 2, \cdots)$ be the length of the ith BRI. We say that a system is *stable* if $L^* < \infty$ exists such that $E[L(i)] < L^*$ for all i. We state a general condition that the arrival process should fulfill so that the system would be stable and show that this condition is closely related to the efficiency of the underlying batch resolution algorithm that is used. Some examples are also given. We consider only the infinite user population model (for a finite user population it is possible to have complete utilization of the channel by using TDMA, for instance).

We start with the following Lemma.

Lemma 3: If for all i there exist T', j^*, $\epsilon > 0$ such that $E[L(i+1)|L(i)=j] < (1-\epsilon)j$ for all $j > j^*$ and $E[L(i+1)|L(i)=j] < T'$ for all $0 \le j \le j^*$, then the system is stable.

Proof: Let $T = \max\{T', E[L(0)]\}$, where $E[L(0)]$ is the expected length of the first BRI (without loss of generality we may assume that $E[L(0)]=1$). Then for all i we have

$$\begin{aligned} E[L(i+1)] &= \sum_{j=0}^{j^*} E[L(i+1)|L(i)=j]P\{L(i)=j\} \\ &\quad + \sum_{j=j^*+1}^{\infty} E[L(i+1)|L(i)=j] \\ &\quad \cdot P\{L(i)=j\} \\ &\le T + (1-\epsilon)E[L(i)]. \end{aligned} \tag{23}$$

The recurrence in (23) implies that

$$\begin{aligned} E[L(i+1)] &< \sum_{l=0}^{i+1} T(1-\epsilon)^l = T\left(1-(1-\epsilon)^{i+2}\right)/\epsilon \\ &\le T/\epsilon = L^* < \infty, \qquad \text{for all } i. \end{aligned} \tag{24}$$

Q.E.D.

Let $N_{t,t+\tau}$ be the number of packets arriving to a system during $(t, t+\tau)$, and let $P(k;t;\tau) = \Pr\{N_{t,t+\tau} = k\}$, $k = 0,1,2,\cdots$. Then the following theorem states a stability condition for a system.

Theorem 8: Let a system operate with a complete resolution algorithm with efficiency E. Then the system is stable if $\tau^*, \delta > 0$ exist such that

$$E\left[N_{t,t+\tau}\right] < (E-\delta)\tau, \qquad \text{for all } t, \tau > \tau^*. \tag{25}$$

Proof: For complete batch resolution algorithms we have that for any $\delta' > 0$ some $B > 0$ exists such that for any k, $L_k < k/E' + B$ where $E' = E - \delta'$ and E is the efficiency of the algorithm. Let $\delta' < \delta$. We then have

$$\begin{aligned} E[L(i+1)|L(i)=j] &\le \sup_t E\left[L_{N_{t,t+\tau}}|\tau = j\right] \\ &= \sup_t \sum_{k=0}^{\infty} L_k P(k;t;\tau) \\ &\le \sup_t \sum_{k=0}^{\infty} (k/E' + B)P(k;t;\tau) \\ &= \sup_t \left\{ \frac{1}{E'} E\left[N_{t,t+\tau}\right] + B \right\} \\ &< (1-\epsilon)j, \qquad \text{for all } j > j^* \end{aligned}$$

where the last inequality follows from (25) with $\epsilon = 0.5(\delta - \delta')/E'$ and $j^* = \max\{\tau^*, B/\epsilon\}$. Similarly, using the fact that if $\tau_1 \le \tau_2$, then $N_{t,t+\tau_1} \le N_{t,t+\tau_2}$, it is easy to show that for $j \le j^*$, $E[L(i+1)|L(i)=j] \le (1-\epsilon)j^* = T'$. Using Lemma 3, the proof then follows.

Theorem 8 states a rather general condition for stability. For stationary processes this condition implies that the system is stable whenever $\Lambda < E$ where $\Lambda = \limsup_{\tau\to\infty} E[N_{t,t+\tau}]/\tau$, or in other words, that the system is stable whenever the average arrival rate is less than the efficiency. However, in general, there is no need to assume stationarity.

Some examples of classes of processes are those with interarrival time distributions, such as Poisson, renewal, deterministic, etc. In addition, the incoming traffic can be bursty. For instance, the number of packets arriving at the system at any arrival point might be a random variable, i.e., with probability $p(l)$ $(l=1,2,\cdots)$ l packets arrive at an arrival point. In these examples, when the interarrival time distribution is not known (such as in a general renewal process) or when the distribution $p(l)$ is not known, then at the beginning of a BRI nothing is known about the batch size k. Here is where the importance of batch resolution algorithms of the kind proposed in this paper becomes apparent.

VII. Discussion

A new class of multiple access algorithms has been presented in this paper. The most important feature of these algorithms is their robustness, i.e., that their parameters are not tailored for any specific arrival process, and therefore, their performance (efficiency) is independent of the specific arrival process into the system. Intuitively, the efficiency of an algorithm is the rate at which packets are successfully transmitted when the system is highly loaded (service rate at heavy traffic). Therefore, it is not surprising that the system is stable whenever the average arrival rate is less than the efficiency.

The algorithms are based on an estimation phase that enables the nodes to estimate quite accurately the conflict multiplicity (batch size). The estimation phase is followed by a resolution phase that exploits the result of the estimation phase to efficiently resolve the conflict. For any given estimate, the algorithms presented in Sections IV and V can be applied. If the estimate of the batch size is asymptotically accurate and the estimation phase is asymptotically short (compared to the resolution phase), then the overall algorithm would yield the same efficiency as our algorithms.

The efficiency of an algorithm is not the sole measure of interest. Other performance measures of interest include the expected length of a BRI for finite batches, the expected delay of a packet for various arrival processes, etc. In the following we compare the performance of our algorithm with the performance of other algorithms when k is finite. To that end we evaluated the ratio $\{k/L_k^1\}$ for $k = 2, 10, 100$ (see Table II) for the modified binary tree (from (1)), for the hybrid algorithm of [8] (from simulation results presented in [8, table 4]), and for our algorithm with the indicated parameters. Clearly, our algorithm outperforms all the others, except the modified binary tree algorithm for very small values of k. Unfortunately, the

TABLE II

k	Modified Binary Tree Algorithm	Hybrid Algorithm of [8]	Our Algorithm $p=0.1$; $\beta=8$
2	0.453	0.374	0.410
10	0.396	0.396	0.407
100	0.383	0.418	0.449

values of L_k^1 for finite values of k for the algorithm in [10] (for $a<2$) are not available. Yet, to see the effect of decreasing a in the algorithm of [10], we computed the expected length of the estimation phase of thatalgorithm for $a=1.5$, 1.1, and 1.01. The respective expected lengths are 4.55, 14.2, and 91.4 (slots) when $k=10$. Assuming that the estimate is completely accurate and that one packet is always transmitted successfully during the estimation phase of [10], we obtain a lower bound on the expected length of the BRI. For $a=1.5$, 1.1, and 1.01, the respective lower bounds on the expected BRI lengths are 28.1, 37.8, and 115 (slots) when $k=10$. This illustrates the main drawback of the estimation procedure of [10], that for any finite k, when $a\to 1$ (to increase the efficiency), the expected length of the estimation phase increases, and this increase cannot be compensated during the resolution phase.

For our algorithm we can compute the expected length of a BRI for any finite k. From Fig. 2 we see that when $p=0.1$, the expected length of a BRI is 24.5 for $k=10$ (even for $\beta=0$). From Theorem 2 we see that for any finite k, the expected length of our estimation phase as a function of p is always bounded. This is not surprising; our estimation procedure is itself a batch resolution algorithm. The drawback of our estimate is that for any finite k, as p gets smaller (to increase the efficiency), the estimate becomes less accurate. However, it is very easy to realize that this will not cause any problems; we can always ignore the information obtained from the estimation phase if we see that $|E|=j$ is small (say 1 or 2).

Finally, we note that for any value of $p>0$ used in our estimation phase, the estimate obtained is asymptotically tight. We also note that the efficiencies of our algorithms are not very sensitive to small variations in p. Therefore, in practice, one can use $p=0.1$ without having considerable deterioration in the efficiency of the algorithm and also having good behavior of the algorithms for small batches.

APPENDIX

Proof of Theorem 3

Our points of departure are the following equations:

$$L_j^r(k)=m\sum_{i=0}^{n}\binom{n}{i}\left(\frac{1}{m}\right)^i\left(1-\frac{1}{m}\right)^{n-i}L_i \qquad \text{(A1a)}$$

$$L^r(k)=E\left[L_j^r(k)\right] \qquad \text{(A1b)}$$

where $n=k-j$, $m=\max\{1,\lceil \alpha j(1-p)/p-\beta\rceil\}$, and $E[\cdot]$ is the expectation operator.

Let $e(\epsilon)$ and $e^c(\epsilon)$ denote the event $\{(p-\epsilon)k\le j\le(p+\epsilon)k\}$ and its complement, respectively. Using Lemma 2, we conclude that for any $\epsilon>0$: $P\{e^c(\epsilon)\}\le p(1-p)/\epsilon^2k$.

The event $e(\epsilon)$ implies the following inequalities:

$$n=(1-p-\epsilon)k\le n\le(1-p+\epsilon)k=\bar{n}. \qquad \text{(A2a)}$$

If $\hat{\delta}=\epsilon(1-p)/p$, then

$$\alpha k(1-p-\hat{\delta})-\beta\le m\le\alpha k(1-p+\hat{\delta})+1. \qquad \text{(A2b)}$$

Using the foregoing, if $\delta=2\hat{\delta}$, then some K_0 ($K_0=\lceil(1/\alpha\hat{\delta})\max\{\beta,1\}\rceil$) exists such that if $k\ge K_0$ and $e(\epsilon)$ holds, then

$$m=\alpha(1-p-\delta)k\le m\le\alpha(1-p+\delta)k=\bar{m}. \qquad \text{(A2c)}$$

Note that $\delta\to 0$ as $\epsilon\to 0$.

Let us denote by $L_j^r(k|e(\epsilon))$ and $L_j^r(k|e^c(\epsilon))$ the average length of the complete resolution procedure given the events $e(\epsilon)$ and $e^c(\epsilon)$, respectively. It is clear that for each $\epsilon>0$,

$$\begin{aligned}\frac{1}{k}L^r(k)&=P\{e(\epsilon)\}E\left[\frac{1}{k}L_j^r(k|e(\epsilon))\right]\\&\quad+P\{e^c(\epsilon)\}E\left[\frac{1}{k}L_j^r(k|e^c(\epsilon))\right]\\&\le 1\cdot\max_{j\in e(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e(\epsilon))\right\}\\&\quad+\frac{p(1-p)}{\epsilon^2k}\max_{j\in e^c(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e^c(\epsilon))\right\}.\end{aligned} \qquad \text{(A3)}$$

It follows that

$$\begin{aligned}\limsup_{k\to\infty}\frac{1}{k}L^r(k)&\le\limsup_{k\to\infty}\max_{j\in e(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e(\epsilon))\right\}\\&\quad+\limsup_{k\to\infty}\frac{p(1-p)}{\epsilon^2k}\max_{j\in e^c(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e^c(\epsilon))\right\}.\end{aligned} \qquad \text{(A4)}$$

Since (A4) holds for all $\epsilon>0$, then

$$\begin{aligned}\limsup_{k\to\infty}\frac{1}{k}L^r(k)&\le\limsup_{\epsilon\to 0}\limsup_{k\to\infty}\max_{j\in e(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e(\epsilon))\right\}\\&\quad+\limsup_{\epsilon\to 0}\limsup_{k\to\infty}\frac{p(1-p)}{\epsilon^2k}\max_{j\in e^c(\epsilon)}\left\{\frac{1}{k}L_j^r(k|e^c(\epsilon))\right\}.\end{aligned} \qquad \text{(A5)}$$

In the following we derive an upper bound for the first term of the right side of (A5). Let us define $\Delta_i=L_i-C_ui+1$. For all j and k using (A1a) given $e(\epsilon)$,

$$\frac{1}{k}L_j^r(k|e(\epsilon))=\frac{m}{k}\sum_{i=0}^{n}\binom{n}{i}\left(\frac{1}{m}\right)^i\left(1-\frac{1}{m}\right)^{n-i}L_i. \qquad \text{(A6)}$$

If $k\ge K_0$,

$$\le\frac{\bar{n}C_u}{k}-\frac{m}{k}+\frac{m}{k}\sum_{i=0}^{3}\binom{n}{i}\left(\frac{1}{m}\right)^i\left(1-\frac{1}{m}\right)^{n-i}\Delta_i.$$

Using $\Delta_0=2$, $\Delta_1=-0.623$, $\Delta_2=0.1683$, and $\Delta_0=0.0157$,

$$\begin{aligned}&\le\frac{1}{k}\left\{\bar{n}C_u-m+m\left(1-\frac{1}{m}\right)^n\Delta_0+n\left(1-\frac{1}{m}\right)^n\Delta_1\right.\\&\quad\left.+\frac{n^2}{2m}\left(1-\frac{1}{m}\right)^{n-2}\Delta_2+\frac{n^3}{6m^2}\left(1-\frac{1}{m}\right)^{n-3}\Delta_3\right\}\\&\le\frac{1}{k}\left\{\bar{n}C_u-m+\bar{m}\left(1-\frac{1}{\bar{m}}\right)^{\bar{n}}\Delta_0+n\left(1-\frac{1}{m}\right)^{\bar{n}}\Delta_1\right.\\&\quad\left.+\frac{\bar{n}^2}{2m}\left(1-\frac{1}{\bar{m}}\right)^{\bar{n}-2}\Delta_2+\frac{\bar{n}^3}{6m^2}\left(1-\frac{1}{\bar{m}}\right)^{\bar{n}-3}\Delta_3\right\}.\end{aligned}$$

Using the equality $\lim_{k\to\infty}(1-(1/ak))^{bk-c}=e^{-b/a}$, and by letting $k\to\infty$,

$$\begin{aligned}\limsup_{k\to\infty}\frac{1}{k}L_j^r(k)&\le(1-p+\epsilon)C_u-\alpha(1-p-\delta)\\&\quad+\frac{(1-p-\delta)(1-p-\epsilon)}{\alpha(1-p+\delta)}\exp\left(-\frac{1-p+\epsilon}{\alpha(1-p-\delta)}\right)\Delta_1\\&\quad+(1-p+\delta)\exp\left(-\frac{1-p-\epsilon}{\alpha(1-p+\delta)}\right)\\&\quad\cdot\left\{\Delta_0+\frac{(1-p+\epsilon)^2}{2(\alpha(1-p-\delta))^2}\Delta_2+\frac{(1-p+\epsilon)^3}{6(\alpha(1-p-\delta))^3}\Delta_3\right\}.\end{aligned}\tag{A7}$$

Letting $\epsilon\to 0$, we conclude that

$$\limsup_{\epsilon\to 0}\limsup_{k\to\infty}E\left[\frac{1}{k}L_j^r(k|e(\epsilon))\right]\le(1-p)C_u-\alpha(1-p)+(1-p)e^{-(1/\alpha)}\sum_{i=0}^{3}\frac{\Delta_i}{\alpha^i i!}.\tag{A8}$$

We turn now to calculate an upper bound for $\limsup_{k\to\infty}(1/k)L_j^r(k|e^c(\epsilon))$. To that end, we use the inequalities $L_i\le C_u i+1$ for $i\ge 0$, $m\le k(1-p)/p+1$, $k-j\le k$ for all j, and obtain that for each j,

$$\begin{aligned}\frac{1}{k}L_j^r(k|e^c(\epsilon))&\le\frac{m}{k}\sum_{i=0}^{n}\binom{n}{i}\left(\frac{1}{m}\right)^i\left(1-\frac{1}{m}\right)^{n-i}(C_u i+1)\\&\le\frac{m}{k}+\frac{n}{k}C_u\le\frac{(1-p)}{p}+1+C_u.\end{aligned}\tag{A9}$$

Now by using Chebyshev's result, $P\{e^c(\epsilon)\}\le p(1-p)/\epsilon^2 k$, we conclude that for each $\epsilon>0$,

$$\limsup_{k\to\infty}P\{e^c(\epsilon)\}E\left[\frac{1}{k}L_j^r(k|e^c(\epsilon))\right]\le\limsup_{k\to\infty}\frac{p(1-p)}{\epsilon^2 k}\left(\frac{1-p}{p}+1+C_u\right)=0.\tag{A10}$$

Substituting the results of (A8) and (A10) into (A5) completes the proof of Theorem 3.

REFERENCES

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

[2] B. S. Tsybakov and V. A. Mikhailov, "Free synchronous packet access in a broadcast channel with feedback," *Probl. Peredach. Inform.*, vol. 14, no. 4, pp. 32–59, Oct.–Dec. 1978.

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

[4] J. L. Massey, "Collision resolution algorithms and random-access communications," Univ. California, Los Angeles, Tech. Rep. UCLA-ENG-8016, Apr. 1980; also in *Multi-User Communications Systems* (CISM Courses and Lectures Series), G. Longo, Ed. New York: Springer-Verlag, 1981, pp. 73–137.

[5] G. Ruget, "Some tools for study of channel-sharing algorithms," in *Multi-User Communications Systems* (CISM Courses and Lectures Series), G. Longo, Ed. New York: Springer-Verlag, 1981, pp. 201–231.

[6] A. G. Greenberg, "Conflict resolution in random access broadcast networks," in *Proc. 14th Ann. ACM Symp. Theory of Computing*, 1982.

[7] M. Hofri, "Stack algorithms for collision-detecting channels and their analysis: A limited survey," Israel Institute of Technology, Department of Computer Science, Tech. Rep. 266, Feb. 1983.

[8] A. G. Greenberg and R. E. Ladner, "Estimating the multiplicities of conflicts in multiple access channels (preliminary report)," *Proc. Foundation Comput. Syst.* (FOCS), pp. 383–391, 1983.

[9] J. Mosely, "An efficient contention resolution algorithm for multiple access channels," Lab. Inform. Decision Syst., Mass. Inst. Technol., Cambridge, Rep. LIDS-TH-918, June 1979.

[10] A. G. Greenberg, P. Flajolet, and R. E. Ladner, "Estimating the multiplicities of conflicts to speed their resolution in multiple access channels," *J. ACM*, vol. 34, no. 2, pp. 289–325, April 1987.