Splitting Protocols in Presence of Capture

MOSHE SIDI, MEMBER, IEEE, AND ISRAEL CIDON

Abstract—**The effect of capture on splitting-type protocols in a slotted ALOHA broadcasting network is investigated. At first, it is assumed that the nodes of the network are divided into two groups, and only packets sent by nodes of one of the groups might be captured. The situation in which the receiver can distinguish between success slots and capture slots and that in which it cannot are both considered. For each of these situations, splitting-type multiple access protocols are described and their performance in terms of achievable throughputs is evaluated. Extensions of these protocols to a general capture model are also discussed.**

I. Introduction

IN THIS PAPER we present and analyze several multiple access protocols suited to time-slotted capture-type channels. These protocols are adapted from the splitting protocol introduced independently by Gallager [1] and by Tsybakov and Mikhailov [7] and subsequently improved by Mosely [2]. The original splitting protocol [1] was devised for collision-type channels where the underlying assumption is that whenever two or more packets are transmitted during the same slot, then neither of them is correctly received at the common receiver. In fact, this assumption provides a lower bound to the performance of real networks, since in many communication systems the stronger of two or more overlapping packets may *capture* the receiver and thus be received without error. As we shall subsequently show, it is possible to improve the performance of the system by taking advantage of the capture effect.

The capture effect has been investigated in [5], [6] for the classic slotted ALOHA random access scheme under the hypothesis that the combined traffic of new and retransmitted packets forms a stationary and independent Poisson process. Collision resolution algorithms with capture that are adapted from the tree algorithm first introduced by Capetanakis [3] have been presented and analyzed in [4].

The splitting protocol makes use of a feedback information channel that tells the nodes of the network, immediately at the end of each slot, what happened during that slot. In the absence of captures, the distinguishable events during a slot are assumed to be: 1) idle slot—no packet is transmitted during the slot; 2) success slot—exactly one packet is transmitted during the slot and therefore correctly received; 3) collision slot—two or more packets are transmitted during the slot and neither of them is correctly received. When packets can be captured, the following additional event is assumed: 4) capture slot—two or more packets are transmitted during the slot and one of them is correctly received. As in [1], we assume that, when events 1), 2), or 3) occur, the receiver broadcasts through the feedback channel LACK, ACK, or NACK, respectively. If 4) occurs we consider two possible situations. In the first, we assume that the receiver can distinguish between success slots and capture slots; in the second, that it cannot. These two situations differ in the meaning of the feedback for event 4), and we will investigate them separately.

To facilitate the exposition of the protocols and to ease their analysis, our detailed presentation and analysis is confined to a simplified model in which the nodes of the network can be divided into two groups, and this division is fixed in time. Only packets transmitted by nodes from one of the groups can be captured at the common receiver. The exact model with its basic assumptions and the description of the two possible feedback information situations is given in Section II. In Section III, we review the basic splitting protocol [1] that is extended in the following sections to protocols for the capture-type channel. In Sections IV and V, we give splitting protocols for the case where the receiver can distinguish between success and capture slots and for the case where it cannot, respectively. The achievable throughputs of these protocols are evaluated in Section VI. In Section VII, we generalize the capture model and describe the corresponding protocols; in Section VIII, we discuss some other extensions.

II. The Basic Model

We consider the accessing of a common receiver by a very large, effectively infinite number of nodes. The forward channel to the receiver is a noiseless time-slotted capture-type common radio channel. The nodes transmit packets of a fixed length, taken as one slot, and are synchronized in the sense that each transmitted packet arrives at the receiver over exactly one slot. The nodes of the network are divided into two groups, one called the dominating group (DG) and the other the nondominating group (NDG). Only packets transmitted by nodes from the DG can be captured at the receiver. The nodes of the DG and the NDG generate new packets according to Poisson processes with rates λ_{DG} and λ_{NDG}, respectively.

Manuscript received July 20, 1984; revised January 3, 1985. This work was supported in part by the National Science Foundation under Grant NSF-ECS-8310698.

M. Sidi was with the Massachusetts Institute of Technology, Cambridge, MA 02139, USA. He is presently with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

I. Cidon is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

0018-9448/85/0300-0295$01.00

The possible events that may occur during each slot in a capture-type channel are: 1) idle slot; no node is transmitting during the slot; 2) success slot; exactly one node (from either the DG or the NDG) is transmitting, in which case its packet is successfully received; 3) collision slot; two or more nodes from the DG with any number of nodes from the NDG, or no node from the DG and at least two nodes from the NDG use the slot, in which case individual transmitted packets cannot be reconstructed at the receiver and must be retransmitted at some later time; 4) capture slot; exactly one node from the DG and one or more nodes from the NDG use the channel during the slot. In case 4) we assume that the receiver captures the packet transmitted by the node from the DG, therefore it is successfully received at the receiver. All other packets that have been sent by nodes from the NDG during the capture slot cannot be reconstructed at the receiver and must be retransmitted at some later time.

The feedback channel from the common receiver is assumed to be a noiseless broadcast channel. Through this channel the receiver tells all nodes of the network, immediately at the end of each slot, what happened during that slot. If either of the above events 1), 2), or 3) occurs, then the receiver broadcasts LACK, ACK, or NACK, respectively. If the above event 4) occurs, i.e. whenever a packet from the DG is captured, we distinguish two possible feedback information situations.

The first of these feedback situations will be called *feedback with capture* (FWC). With FWC, the receiver is able to distinguish between capture slots and success slots, i.e. it is able to detect that it has received correctly a packet from the DG, though at least one packet from the NDG has been simultaneously transmitted. In this case the receiver broadcasts CAPT to all nodes in the network.

The second of these feedback information situations will be called *feedback without capture* (FWOC), With FWOC, the receiver cannot distinguish between capture slots and success slots, i.e. it cannot decide whether a correctly received packet has been the only transmitted packet or whether other packets have also been transmitted. In this case, the receiver could broadcast ACK to all nodes in the network. However, had a capture occurred, such an ACK might confuse the nodes from the NDG that have been transmitting, since they will not know that their packets have not been successfully received. Therefore, with FWOC, we assume that each node adds a group identity number (in this case one bit) at the head of a transmitted packet. This added bit is zero or one if the node belongs to the DG or to the NDG, respectively. Consequently, with FWOC, the receiver broadcasts ACK_0 or ACK_1 when it receives correctly a packet from a node that belongs to the DG or to the NDG, respectively. Nonetheless, note that FWOC feedback is less informative than FWC feedback because, with FWOC, the receipt of ACK_0 gives no information regarding the presense or absence of packets from the NDG.

In all cases, we assume that propagation delay is negligible, so that the feedback information for a certain slot can be used to determine who should transmit in the very next slot.

III. The Basic Splitting Protocol

An understanding of Gallager's basic splitting protocol [1] is necessary to the understanding of splitting protocols for capture-type channels. In the basic splitting protocol, each node of the system keeps track of an interval of time in the past which shall be called a window. The left and right boundaries of the window will be denoted by A and B, respectively. During each slot, all packets that arrive during the current window are transmitted. The system may reside in one of three states, S_0, S_1, S_2, depending on what knowledge the nodes have regarding the current window. Recall that here the only feedbacks possible are LACK, ACK, or NACK, and that the same window is tracked by all nodes.

In state S_0, the system has no knowledge regarding the number of packets that arrived within the current window. For LACK or ACK received from a slot when the system is in state S_0, the system remains in S_0 but slides the window forward. The new window size will be chosen between μ and $T_c - B$, whichever is smaller ($A \leftarrow B$, $B \leftarrow \min(\mu + B, T_c)$). Here μ is a constant that is chosen so that some performance measure is optimized, and T_c is the current time. For NACK feedback, the system enters state S_2 with the knowledge that the window contains at least two packets. In this transition, the window is partitioned into a left part and a right part according to some parameter p (that is again chosen so that some performance measure is optimized), and the left part becomes the new window ($B \leftarrow A + (B - A)p$).

If the system is in state S_2, then LACK or NACK feedback maintains the system in state S_2. For NACK feedback, the new window is obtained as in the transition from S_0 to S_2. For LACK feedback, the right part of the previous window is known to contain at least two packets, therefore it is partitioned and a new window is chosen as in the transition from S_0 to S_2 ($A \leftarrow B$; $B \leftarrow B + (B - A)(1 - p)$). An ACK feedback allows the system to transit from state S_2 to state S_1, since the nodes now know that the right part of the current window, which becomes the new window ($A \leftarrow B$; $B \leftarrow [B - A(1 - p)]/p$), contains at least one packet.

Finally, from state S_1, an ACK or NACK feedback causes the system to transit to S_0 or S_2, respectively. In the former case, the new window is obtained by sliding the window forward in time and expanding it to length μ or $T_c - B$, whichever is smaller. In the latter case, the current window is partitioned and the new window is chosen as in the transition from S_0 to S_2.

IV. Splitting Protocols for Feedback with Capture

In feedback with capture (FWC), the receiver can distinguish between success slots and capture slots. Therefore the possible feedback messages are LACK, ACK, NACK

and CAPT. We shall present two splitting protocols for FWC that are adapted from the basic splitting protocol described in the previous section. The principal difference between the two protocols is that in the first (Protocol 1) at most one packet can be captured between two successive visits to state S_0, while in the second (Protocol 2) the number of captured packets between such two successive visits can also be two.

In both protocols, nodes from different groups (the DG and the NDG) keep track of different windows. Let (A_1, B_1) and (A_2, B_2) be the windows of DG nodes and NDG nodes, respectively. For both protocols, the nodes perform the basic splitting protocol for LACK, ACK or NACK, the DG nodes using their own parameters, p_1, μ_1 and the NDG nodes using their own parameters, p_2, μ_2. For both protocols, when the system is in state S_0 or S_1 and a capture occurs, then the corresponding window of the NDG nodes is known to contain at least one packet; therefore, the DG nodes wait until the window of the NDG nodes involved in the capture is resolved (a window is said to be resolved whenever the system reenters state S_0). This is achieved by choosing a zero length window for the DG nodes with no change in the window for the NDG nodes, while the system transits to state S_1.

The two protocols differ whenever a capture occurs while the system is in state S_2. Note that, in this case, it is known that the current window of the NDG nodes contains at least one packet, and that the right part of the previous nonzero window of the DG nodes also contains at least one packet. A reasonable approach in this case is to resolve separately the two windows known to contain at least one packet, and then to choose fresh ones. This is what is done in Protocol 1. According to this protocol, after a capture while the system is in state S_2, the NDG window involved in the capture is first resolved while the DG nodes wait. Then the right part of the previous nonzero window of the DG nodes is resolved while the NDG nodes wait. This is achieved by first choosing a window of zero length for the DG nodes, raising a flag, and resolving the window of the NDG. When this window is resolved, a window of zero length is chosen for the NDG, the flag is lowered, and the right part of the previous nonzero window of the DG is resolved. Consequently, at most one packet can be captured between two successive visits to state S_0.

Obviously, more information is extracted from a capture than from a simple successful transmission. Thus, one expects that a protocol that allows for more than one capture between two successive visits to state S_0 will perform better. This idea is used in Protocol 2. According to this protocol, if a capture occurs while the system is in state S_2, then the NDG window involved in the capture is first resolved while DG nodes wait, as in Protocol 1. After this window is resolved, DG nodes choose the right part of their previous nonzero window as their new window, as before, but now NDG nodes do not wait but choose a completely fresh window of length μ_2' or $T_c - B_2$, whichever is smaller, and continue the protocol with partitioning parameter p_2'. In this case, the system transits from state S_2 to state S_1. If a capture occurs, then the protocol continues as described above. However, if a collision occurs, then it is known that the window of the DG nodes contains at least two packets and thus no information has been gained regarding the NDG window, which can then be considered as fresh. Therefore, the protocol continues as before except that the NDG window is reduced to zero length.

V. Splitting Protocols for Feedback without Capture

With feedback without capture (FWOC), the receiver cannot distinguish between capture and success slots. Therefore it always broadcasts either ACK_0 or ACK_1 after receiving correctly a packet from a node from the DG or the NDG, respectively. We now present two splitting protocols for FWOC. In principal, in the first protocol (Protocol 3), each time a nonempty DG window is resolved, DG nodes wait until the NDG nodes resolve their current window. The purpose is to allow the NDG nodes that were involved in a capture slot to transmit before a new window for the DG nodes is chosen. In the other and more efficient protocol (Protocol 4), the DG nodes choose a new window immediately after resolving their previous window.

In both protocols (Protocols 3 and 4), nodes of the DG and the NDG perform their basic window protocol except when a packet from a DG node is successfully transmitted (i.e., when ACK_0 is received). Note that in this case it is possible that the window of the NDG nodes is not empty. When ACK_0 is received while the system is in state S_2, then since nothing is known regarding the window of the NDG nodes, this window is extended in both protocols so that it contains the right part of the previous window. When ACK_0 is received while the system is in either state S_0 or S_1, then in Protocol 3 the NDG nodes continue their protocol with no immediate change in the window while the DG nodes wait (by reducing their window to zero length) until the window of the NDG nodes is resolved. It is clear that this is not the best we can do, because in this case no further statistical knowledge was gained about the current NDG window. Therefore we can improve the performance of the protocol simply by choosing completely new windows for both the DG and the NDG nodes. This is exactly what is done in Protocol 4. Notice that in Protocol 4 packets from NDG nodes might be delayed for long periods if several successive captures occur.

VI. Performance Evaluation

In this section, we assume that the arrival processes of new packets to the nodes are Poisson. We determine the set of achievable arrival rates λ_{DG} and λ_{NDG} of new packets to nodes of the DG and NDG nodes, respectively, for which the system is stable. These rates clearly depend on the fresh window lengths μ_1 and μ_2 and on the partitioning parameters p_1 and p_2 (in Protocol 2, also on μ_2' and p_2'). In particular, we obtain the optimal fresh window lengths μ_1^* and μ_2^* and partitioning parameters p_1^* and p_2^*

so that the total throughput of the system $\lambda_T = \lambda_{DG} + \lambda_{NDG}$ is maximized.

We first define some notation (here $\boldsymbol{p} = (p_1, p_2)$, and n and k are the number of packets contained in the current windows of the DG and NDG nodes, respectively).

$L_{n,k}^{DG}(\boldsymbol{p}) \triangleq$ the average number of successfully transmitted packets by nodes of the DG until the next visit to state S_0, given n and k.

$L_{n,k}^{NDG}(\boldsymbol{p}) \triangleq$ the average number of successfully transmitted packets by nodes of the NDG until the next visit to state S_0, given n and k.

$L_{n,k}(\boldsymbol{p}) \triangleq L_{n,k}^{DG}(\boldsymbol{p}) + L_{n,k}^{NDG}(\boldsymbol{p})$.

$M_{n,k}(\boldsymbol{p}) \triangleq$ the average number of slots until the next visit to state S_0, given n and k.

In the Appendix, we present the equations from which these quantities can be calculated recursively.

Since the arrival processes of new packets to nodes of the DG and the NDG are Poisson with rates λ_{DG} and λ_{NDG}, respectively, we have that the average number of successfully transmitted packets from DG nodes and NDG nodes between two successive visits to state S_0 is given by

$$L^{DG}(\boldsymbol{p}, \boldsymbol{x}) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} L_{i,j}^{DG}(\boldsymbol{p}) \frac{e^{-(x_1+x_2)} x_1^i x_2^j}{i!j!}$$

$$L^{NDG}(\boldsymbol{p}, \boldsymbol{x}) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} L_{i,j}^{NDG}(\boldsymbol{p}) \frac{e^{-(x_1+x_2)} x_1^i x_2^j}{i!j!}$$

where

$$x_1 \triangleq \lambda_{DG}\mu_1 \qquad x_2 = \lambda_{NDG}\mu_2$$

and

$$\boldsymbol{x} \triangleq (x_1, x_2).$$

The total average number of successful transmissions between two successive visits to S_0 is

$$L(\boldsymbol{p}, \boldsymbol{x}) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} L_{i,j}(\boldsymbol{p}) \frac{e^{-(x_1+x_2)} x_1^i x_2^j}{i!j!}.$$

The average number of slots elapsed between two successive visits to S_0 is

$$M(\boldsymbol{p}, \boldsymbol{x}) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} M_{i,j}(\boldsymbol{p}) \frac{e^{-(x_1+x_2)} x_1^i x_2^j}{i!j!}.$$

The throughputs of the system are finally given by

$$T^{DG}(\boldsymbol{x}, \boldsymbol{p}) = \frac{L^{DG}(\boldsymbol{x}, \boldsymbol{p})}{M(\boldsymbol{x}, \boldsymbol{p})}$$

$$T^{NDG}(\boldsymbol{x}, \boldsymbol{p}) = \frac{L^{NDG}(\boldsymbol{x}, \boldsymbol{p})}{M(\boldsymbol{x}, \boldsymbol{p})}.$$

Clearly, the throughputs depend on the partitioning parameters, p_1 and p_2, and on the lengths of new windows, μ_1 and μ_2, through the quantities x_1 and x_2. As for the basic splitting protocol [1], the throughputs of the system are not very sensitive to the partitioning parameters p_1 and p_2 and, therefore, in the following we choose $p_1 = p_2 = 1/2$.

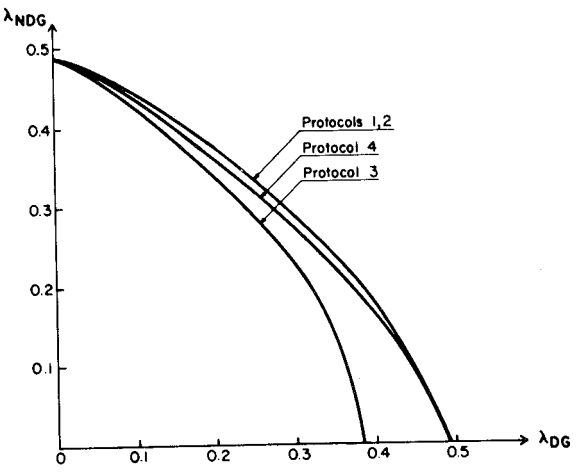

Fig. 1. Regions of achievable throughputs.

In Fig. 1 the region of achievable arrival rates for which the system is stable is under the depicted line for each protocol. Protocol 2 is only slightly better than Protocol 1 (the difference is not noticeable in the figure). Protocol 3 behaves quite badly for high arrival rates to DG nodes. The reason is that after each ACK_0 at least one slot is dedicated for NDG nodes even if no node from this group has a packet for transmission. Notice that the basic window protocol (without capture) corresponds to $\lambda_{DG} = 0$ or $\lambda_{NDG} = 0$ in Fig. 1, except for Protocol 3. From Fig. 1, we can conclude that Protocols 1 (because of its simplicity compared to Protocol 2) and 4 are the better choices for FWC and FWOC, respectively.

It is interesting to find the values $\mu_1^*, \mu_2^*, p_1^*, p_2^*, x_1^*, x_2^*$ that maximize the total throughput T. The values of these parameters together with the corresponding maximal throughputs appear in Table I. For Protocol 2, the optimizing values of μ_2' and p_2' are 1.52 and 0.49, respectively.

TABLE I
MAXIMUM TOTAL THROUGHPUTS AND CORRESPONDING PARAMETERS

Protocol	μ_1^*	μ_2^*	p_1^*	p_2^*	λ_{DG}^*	λ_{NDG}^*	$\lambda_{DG}^* + \lambda_{NDG}^*$
1	2.51	2.61	0.48	0.49	0.295	0.293	0.588
2	2.54	2.60	0.48	0.49	0.293	0.296	0.589
3	2.95	3.17	0.47	0.49	0.190	0.350	0.540
4	2.10	4.01	0.47	0.49	0.329	0.244	0.573

Notice that for FWC the maximal total throughput is obtained for almost the same arrival rates to DG nodes and NDG nodes, and that there is almost no difference between the optimal parameters and the performance of Protocols 1 and 2. Protocol 3 favors higher arrival rates to NDG nodes, since successful transmissions from DG nodes that are not caused by captures lead to wasted slots. Protocol 4 favors higher arrival rates to DG nodes, because in FWOC DG nodes have strong priority over NDG nodes.

In all the above calculations, we truncated the infinite sums at $i = 16$ and $j = 16$; the contributions from the other terms were negligible.

VII. Generalization

In this section, we generalize the basic model described in Section II and show how to apply the various splitting protocols to this generalized model. In the general model, we assume that the nodes of the network are divided into a finite number of groups, each of which may contain any number (finite or infinite) of nodes. This division is done so that a capture never occurs within nodes of the same group. The groups are uniquely (yet arbitrarily) identified by a group identity number. Each transmitted packet carries the group identity number of its originator node.

The possible events that may occur during each slot are similar to those described in Section II: 1) idle slot; 2) success slot; 3) collision slot—two or more nodes use the channel but none of the individual transmitted packets can be reconstructed at the receiver; 4) capture slot—two or more nodes use the channel, as in 3), but one of the packets capture the channel and therefore is successfully received at the receiver, although other packets must be retransmitted at some later time. The collection of nodes that must retransmit their packets after a capture slot will be called a *capture set*. We note that each packet from the capture set carries a group identity number that differs from the one carried by the successfully received packet.

We consider here both the FWC and the FWOC as in Section II, and we assume that CAPT and ACK messages contain the group identity number of the successfully received packet.

Essentially, we can still use the protocols that were developed in Sections IV and V for the basic model, except that now, during each slot, the nodes in the group from which a packet was successfully received follow the earlier protocol for DG nodes, while all other nodes follow the earlier protocol for NDG nodes. Note that, in Protocols 1 and 3, whenever the DG nodes should wait for resolution of the NDG window in the basic model, nodes in the group from which a packet was successfully received must wait until the groups in the capture set have resolved their windows.

VIII. Discussion

We have investigated the effect of capture on the basic splitting protocol [1], [2]. We have considered both the situations where the receiver can distinguish between success slots and capture slots and where it cannot. For each of these cases, we described splitting-type multiple-access protocols and evaluated their performance. As could be expected, we have shown that the performance of the system is significantly improved when packets can be captured.

Protocols have been suggested for a rather general capture model. The detailed exposition of the protocols and their analysis have been confined to the case where the nodes of the system are divided into two groups and only nodes of one group (DG nodes) could be captured. Moreover, we assumed that a capture event does not depend on the number of NDG nodes transmitting along with a node from the DG.

A more realistic assumption is that a packet transmitted by a node from the DG is captured only if fewer than K NDG nodes are transmitting at the same time. In some sense, this K corresponds to the relative powers of DG and NDG nodes. $K = 1$ corresponds to the case of no capture. The protocols that we presented earlier can, with very slight modification, be used in this case as well and their performance can be evaluated using the same techniques. It turns out, for example, that for FWC using Protocol 1 and $K = 2$, the performance of the system (in terms of achievable throughputs) is degraded by only about 2 percent compared with the case of very large (effectively infinite) K. This behavior is due to the fact that the windows chosen by the nodes will very rarely contain a large number of packets. Consequently, provided $K \geq 2$, the behavior of the system is almost independent of K.

An interesting special case of the general model considered in Section VII is to assume that the nodes of the system are divided into N priority groups, instead of only two groups as in the basic model. Let the N groups of the system be denoted by $A_1, A_2, \cdots, A_N$. We say that group A_i dominates group A_j whenever $i > j$. The domination has the following sense. If a single packet originated from A_i and if an arbitrary number of packets originated from groups dominated by A_i are transmitted during the same slot, then the common receiver will capture the packet originated at A_i. It is expected that as N increases the performance of the system will be improved and, for very large N, the utilization of the channel will be almost perfect. A first order approximation shows that as N gets large the total throughput increases at least as fast as $1 - C/\sqrt{N}$ for some constant C.

Acknowledgment

The first author would like to thank the Rothschild Foundation and Prof. R. G. Gallager for the opportunity to spend a year at the Massachusetts Institute of Technology in research on random-access problems.

Appendix

Let $L_{n,k}^{DG}$, $L_{n,k}^{NDG}$, $L_{n,k}$ and $M_{n,k}$ be as defined in Section VI. Let

$$P_i^1(n) = \binom{n}{i} p_1^i (1 - p_1)^{n-i}$$

$$P_i^2(n) = \binom{n}{i} p_2^i (1 - p_2)^{n-i}.$$

In the following, we present the equations that are used for recursive computation of the quantities of interest. Most of these equations are self-explanatory. To help the reader, we give detailed explanations for one of the following equations, (A6). The other equations are similarly obtained.

Protocol 1

$$L_{0,0} = 0 \tag{A1}$$

$$L_{1,0} = L_{0,1} = 1 \tag{A2}$$

$$L_{n,0} = P_0^1(n)\,L_{n,0} + P_1^1(n)(1 + L_{n-1,0}) + \sum_{i=2}^{n} P_i^1(n)\,L_{i,0}, \qquad n \geq 2 \quad \text{(A3)}$$

$$L_{0,k} = P_0^2(k)\,L_{0,k} + P_1^2(k)(1 + L_{0,k-1}) + \sum_{i=2}^{k} P_i^2(k)\,L_{0,i}, \qquad k \geq 2 \quad \text{(A4)}$$

$$L_{1,k} = 1 + L_{0,k}, \qquad k \geq 1 \quad \text{(A5)}$$

$$\begin{aligned} L_{n,k} = {} & P_0^1(n)\Big[P_0^2(k)\,L_{n,k} + P_1^2(k)(1 + L_{n,k-1}) + \sum_{i=2}^{k} P_i^2(k)\,L_{0,i}\Big] \\ & + P_1^1(n)\Big[P_0^2(k)(1 + L_{n-1,k}) + \sum_{i=1}^{k} P_i^2(k)(1 + L_{n-1,0} + L_{0,i})\Big] \\ & + \sum_{i=2}^{n}\sum_{j=0}^{k} P_i^1(n)\,P_j^2(k)\,L_{i,j}, \qquad n \geq 2, k \geq 1. \end{aligned} \quad \text{(A6)}$$

Explanation of (A6): Recall that $L_{n,k}$ is the average number of successfully transmitted packets (either from the DG or NDG nodes) until the next visit to state S_0, given that the current windows contain n DG packets and k NDG packets. Therefore, for $n \geq 2, k \geq 1$, the current windows result in NACK. Consequently, the windows of the corresponding groups are partitioned into left and right parts.

If the left part of the DG window contains no packets ($P_0^1(n)$), then three cases are possible. 1) The left part of the NDG window contains no packets ($P_0^2(k)$), in which case we are back in the same situation. 2) The left part of the NDG window contains exactly one packet ($P_1^2(k)$), in which case this packet is successfully transmitted and then the right windows with n DG packets and $k-1$ NDG packets are transmitted. 3) The left part of the NDG window contains at least two packets ($P_i^2(k), i \geq 2$), in which case we are left with a collision that has to be resolved, so according to Protocol 1 we continue to resolve the left parts with no DG packets and k NDG packets.

If the left part of the DG window contains exactly one packet ($P_1^1(n)$), then we have two cases. 1) The left part of the NDG window is empty ($P_0^2(k)$), in which case the DG packet is transmitted successfully and then the right parts with $n-1$ DG packets and k NDG packets are resolved. 2) The left part of the NDG window contains $i \geq 1$ packets ($P_i^2(k), i \geq 1$), in which case a capture occurs and therefore we resolve separately the left part of NDG window (with i packets) and the right part of the DG window (with $n-1$ packets).

Finally, if the left part of the DG window contains $i \geq 2$ packets ($P_i^1(n), i \geq 2$), then we have another collision and only the left parts are resolved; thus, if the left part of the NDG window contains j packets ($P_j^2(k)$), we will have on the average $L_{i,j}$ successful transmissions.

$$L_{n,0}^{DG} = L_{n,0}, \qquad n \geq 0 \quad \text{(A7)}$$

$$L_{0,k}^{DG} = 0, \qquad k \geq 0 \quad \text{(A8)}$$

$$L_{1,k}^{DG} = 1, \qquad k \geq 0 \quad \text{(A9)}$$

$$\begin{aligned} L_{n,k}^{DG} = {} & P_0^1(n)\big[P_0^2(k)\,L_{n,k}^{DG} + P_1^2(k)\big(1 + L_{n,k-1}^{DG}\big)\big] \\ & + P_1^1(n)\Big[P_0^2(k)\big(1 + L_{n-1,k}^{DG}\big) + \sum_{i=1}^{k} P_i^2(k)\big(1 + L_{n-1,0}^{DG}\big)\Big] \\ & + \sum_{i=2}^{n}\sum_{j=0}^{k} P_i^1(n)\,P_j^2(k)\,L_{i,j}^{DG}, \qquad n \geq 2, k \geq 1 \end{aligned} \quad \text{(A10)}$$

$$L_{n,k}^{NDG} = L_{n,k} - L_{n,k}^{DG}, \qquad n \geq 0, k \geq 0 \quad \text{(A11)}$$

$$M_{0,0} = M_{0,1} = M_{1,0} = 1 \quad \text{(A12)}$$

$$M_{n,0} = 1 + P_0^1(n)\,M_{n,0} + P_1^1(n)(1 + M_{n-1,0}) + \sum_{i=2}^{n} P_i^1(n)\,M_{i,0}, \qquad n \geq 2 \quad \text{(A13)}$$

$$M_{0,k} = 1 + P_0^2(k)\,M_{0,k} + P_1^2(n)(1 + M_{0,k-1}) + \sum_{i=2}^{k} P_i^2(k)\,M_{0,i}, \qquad k \geq 2 \quad \text{(A14)}$$

$$M_{1,k} = 1 + M_{0,k}, \qquad k \geq 1 \quad \text{(A15)}$$

$$\begin{aligned} M_{n,k} = {} & 1 + P_0^1(n)\Big[P_0^2(k)\,M_{n,k} + P_1^2(k)(1 + M_{n,k-1}) + \sum_{i=2}^{k} P_i^2(k)\,M_{0,i}\Big] \\ & + P_1^1(n)\Big[P_0^2(k)(1 + M_{n-1,k}) + \sum_{i=1}^{k} P_i^2(k)(1 + M_{n-1,0} + M_{0,i})\Big] \\ & + \sum_{i=2}^{n}\sum_{j=0}^{k} P_i^1(n)\,P_j^2(k)\,M_{i,j}, \qquad n \geq 2, k \geq 1. \end{aligned} \quad \text{(A16)}$$

Clearly, all the quantities, $L_{n,k}^{DG}, L_{n,k}^{NDG}, L_{n,k}, M_{n,k}$ can be recursively computed from (A1)–(A16).

Protocol 2

The same equations (A1)–(A16) govern the system with Protocol 2 except that, in (A6) and (A16) when $n = 2$, the quantities $L_{n-1,0}$ and $M_{n-1,0}$ should be replaced by the quantities $\tilde{L}$ and $\tilde{M}$ that are subsequently calculated. Let

$$Q_i(n) = \binom{n}{i} p_2'^{i}(1 - p_2')^{n-i}.$$

Then,

$$\tilde{L}_1 = 1$$

$$\tilde{L}_n = Q_0(n)\,\tilde{L}_n + Q_1(n)(1 + \tilde{L}_{n-1}) + \sum_{i=2}^{n} Q_i(n)\,\tilde{L}_i, \qquad n \geq 2$$

$$\tilde{M}_1 = 1$$

$$\tilde{M}_n = 1 + Q_0(n)\,\tilde{M}_n + Q_1(n)(1 + \tilde{M}_{n-1}) + \sum_{i=2}^{n} Q_i(n)\,\tilde{M}_i, \qquad n \geq 2.$$

Now,

$$\tilde{L} = 1 + \sum_{i=1}^{\infty} \tilde{L}_i \frac{e^{-x}x^i}{i!}$$

$$\tilde{M} = 1 + \sum_{i=1}^{\infty} \tilde{M}_i \frac{e^{-x}x^i}{i!}$$

where $x = \lambda_{\text{NDG}}\mu_2'\pi$. In the next two protocols, we give only the equations for $L_{n,k}$ and $M_{n,k}$. The equations for $L_{n,k}^{\text{DG}}$ and $L_{n,k}^{\text{NDG}}$ can be obtained similarly.

Protocol 3

$L_{n,k}$, for $n \geq 0, k = 0,1$ and $n = 0, k \geq 0$, is calculated as in (A1)–(A5) and, for $n \geq 2, k \geq 1$,

$$L_{n,k} = P_0^1(n)\left[P_0^2(k)\,L_{n,k} + P_1^2(k)(1 + L_{n,k-1}) + \sum_{i=2}^{k} P_i(k)\,L_{0,i}\right] + P_1^1(n)(1 + L_{n-1,k}) + \sum_{i=2}^{n}\sum_{j=0}^{k} P_i^1(n)\,P_j^2(k)\,L_{i,j}. \tag{A17}$$

$M_{n,k}$, for $n = 0,1, k \geq 0$, is calculated as in (A12), (A14), and (A15). For $n \geq 1, k = 0$, we have

$$\tilde{M}_{n,0} = 1 + P_0^1(n)\,\tilde{M}_{n,0} + P_1^1(n)\left(1 + \tilde{M}_{n-1,0}\right) + \sum_{i=2}^{n} P_i^1(n)\,\tilde{M}_{i,0}, \qquad n \geq 2$$

$$M_{n,0} = 1 + \tilde{M}_{n,0}$$

and, for $n \geq 2, k \geq 1$, we have

$$M_{n,k} = 1 + P_0^1(n)\left[P_0^2(k)\,M_{n,k} + P_1^2(k)(1 + M_{n,k-1}) + \sum_{i=2}^{k} P_n(k)\,M_{0,i}\right] + P_1^1(k)(1 + M_{n-1,k}) + \sum_{i=2}^{n}\sum_{j=0}^{k} P_i^1(k)\,P_j^2(k)\,M_{i,j}. \tag{A18}$$

Protocol 4

$L_{n,k}$ for $n \geq 0, k = 0$ and $n = 0, k \geq 0$, is calculated as in (A1)–(A4).

$$L_{1,k} = 1, \qquad k \geq 1$$

and, for $n \geq 2, k \geq 1$, the recursion (A17) is used.

$M_{n,k}$, for $n \geq 0, k = 0$ and $n = 0, k \geq 0$, is calculated as in (A12)–(A14).

$$M_{1,k} = 1, \qquad k \geq 1$$

and, for $n \geq 2, k \geq 1$, the recursion (A18) is used.

References

[1] R. G. Gallager, "Conflict resolution in random access broadcast network," in *Proc. AFOSR Workshop Commun. Theory Appl.*, Provincetown, MA, Sept. 17–20, 1978, pp. 74–76.

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

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

[4] I. Cidon and M. Sidi, "The effect of capture on collision resolution algorithms," EE Pub. No. 454, Technion–Israel Institute of Technology, Haifa, Israel, July 1983.

[5] J. J. Metzner, "On improving utilization in ALOHA networks," *IEEE Trans. Commun.*, vol. COM-24, pp. 447–448, Apr. 1976.

[6] N. Abramson, "The throughput of packet broadcasting channels," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 117–128, Jan. 1977.

[7] B. S. Tsybakov and V. A. Mikhailov, "Random multiple-packet access: Part and try algorithm," Problemi Peredachi Informatsii, vol. 16, no. 4, pp. 65–79, 1980.