Multiple Access for Multiple Destinations in Ad Hoc Networks

Jeffrey E. Wieselthier, Gam D. Nguyen, Anthony Ephremides

MULTIPLE ACCESS FOR MULTIPLE DESTINATIONS IN AD HOC NETWORKS

Jeffrey E. Wieselthier
Code 5521
Information Technology Division
Naval Research Laboratory
Washington, DC 20375
wieselthier@itd.nrl.navy.mil

Gam D. Nguyen
Code 5521
Information Technology Division
Naval Research Laboratory
Washington, DC 20375
nguyen@itd.nrl.navy.mil

Anthony Ephremides
Electrical and Computer Eng. Dept.
and Institute for Systems Research
University of Maryland
College Park, MD 20742
tony@eng.umd.edu

ABSTRACT

In this paper, we address the medium-access control (MAC) problem in systems with multiple destinations, a problem that is crucial to the development of ad hoc networks. We focus on collision-resolution algorithms, specifically on First-Come, First-Serve (FCFS) splitting algorithms. We illustrate the difficulties that arise because of the interference between two populations of nodes that share a common channel, the most serious consequence of which is the possibility of deadlock that occurs when a well-known improvement for the single-destination case is used. Experimental results verify that the straightforward use of the FCFS algorithm in two-destination environments (without using this improvement) leads to deadlock-free performance. We show how performance can be improved by applying a time-division mechanism to separate interfering groups of nodes.

1. INTRODUCTION

One of the stumbling blocks to the implementation of ad hoc networks is the absence of medium-access control (MAC) schemes that are suitable for multihop networks. Throughout the long history of multiple-access research, virtually all studies of this problem have been limited to the case of a single destination. Although this framework was appropriate to address channel access problems that are characteristic of many cellular and satellite-based systems, the ad hoc networking environment presents challenges to the MAC problem that do not arise in the single-destination case. Several papers on routing in ad hoc networks include ideas and suggestions for handling the MAC issue, but do not examine the fundamental limitations of MAC in ad hoc networks (see e.g., [1], which addresses many of the cross-layer interrelationships in ad hoc networks).

In this paper, we formulate the two-destination multiple-access problem, and present performance results relating to the throughput that can be achieved. We consider a scenario in which some nodes are within communication range of only one of these destinations, others are within range of the other, and some are within range of both (but each of their transmissions are intended for only one of the two destinations). We discuss the complications introduced by multiple destinations, most significantly the impact on the channel feedback process and the resulting impact on algorithm operation and performance. We focus on the fundamental problem of MAC in slotted systems; thus, we do not address carrier-sense multiple access (CSMA) -based approaches.[1]

We consider two classes of channel access protocols, namely Slotted Aloha and splitting (collision-resolution) algorithms. In both cases, we illustrate the dependence of performance on the fraction of nodes that can communicate directly with both destinations. For Slotted Aloha, we evaluate the performance of uncontrolled systems. Our primary focus in this paper is on splitting algorithms for two or more destinations. We study the performance of the standard First-Come, First-Serve (FCFS) splitting algorithm [3]. We illustrate the difficulties that are introduced by the need to resolve collisions among two populations that interfere with each other, and specifically show how an improvement that was developed for the single-destination case results in deadlock in the two-destination case. Finally, we propose improved versions that incorporate the information structures that are associated with the two-destination problem.

2. PROBLEM DEFINITION

Figure 1 shows a network with two destinations (D1 and D2) and a number of nodes that are within communication range of them (assuming the same fixed communication range for all nodes and the use of omnidirectional antennas). We define Group 1 to be the set of nodes within communication range of only D1; similarly Group 2 is the set of nodes within range of only D2. Nodes that are within range of both D1 and D2 (i.e., the intersection of the two circular regions) are said to be in Group 3. We use a simple interference model in which a node can cause interference only if it is within communication range, i.e., the combined effect of transmissions from multiple nodes outside the communication range is not considered.

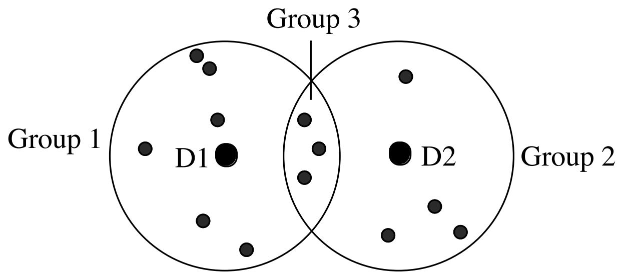

Fig. 1 — Multiple access in two overlapping regions.

Nodes in Group 1 randomly generate packets for transmission to D1; similarly nodes in Group 2 randomly generate packets for transmission to D2. Nodes in Group

This work was supported by the Office of Naval Research.

[1] Some issues associated with the use of IEEE 802.11 (which is based on CSMA with collision avoidance) in ad hoc networks are addressed in [2].

3 randomly generate packets for transmission to either D1 or D2 with equal probability; once a destination is chosen for a particular packet, the packet must be delivered to that destination (i.e., no reward is received for successful reception at the wrong destination).

Since packets from Group 3 reach both destinations, they are potentially a source of collisions at both the intended and non-intended destination. The challenge presented by the two-destination problem is to correctly model the impact of collisions caused by packets intended for another destination, and to develop channel-access protocols that are suitable for use in such environments.

We make the common assumption of an infinite population of unbuffered users in each group (e.g., as in [3]). We define

f = fraction of total traffic generated by nodes in Group 3.

Thus, when $f = 0$, Group 3 is empty and the collision-resolution process can proceed independently for the two destinations. At the other extreme, when $f = 1$, all packets reach both destinations.

The packets are classified into four types as follows, according to their intended destinations and group membership. Packets that are generated by Group 1's nodes are called *type-1 packets*; thus, type-1 packets are intended for D1 only, and never reach D2. Similarly, packets generated by Group 2's nodes are called *type-2 packets*. Recall that packets generated by Group 3's nodes are intended for either D1 or D2, and these packets will reach both D1 and D2. Packets generated by Group 3's nodes that are intended for D1 are called *type-31 packets*, while packets intended for D2 are called *type-32 packets*. These packets are generated according to a random (e.g., Poisson) process.

Each destination broadcasts ternary feedback information (i.e., success, collision, or empty slot) immediately following each slot. We assume that nodes in Group 1 and Group 3 receive D1's feedback, and that nodes in Group 2 and Group 3 receive D2's feedback. However, nodes in Group 1 (Group 2) do not receive feedback from D2 (D1).

3. SLOTTED ALOHA WITH TWO DESTINATIONS

We consider uncontrolled Slotted Aloha, where backlogged users retransmit with probability arbitrarily set to the fixed value of 0.01 in each slot.[2] When $f = 0$, the two regions are disjoint and the problem reduces to that of two independent Aloha systems. Our simulation results provide a maximum throughput of approximately $2/e$ = 0.736 packets per slot, or equivalently $1/e$ per destination, as expected. When $f = 0.4$, and thus 40% of the users are in Group 3, the maximum throughput per destination is approximately 0.26. Of course, such uncontrolled Aloha systems with infinite populations are unstable. They can be stabilized by choosing the retransmission probability as a function of estimates of the number of backlogged users. However, better (and stable) performance can be achieved by using splitting algorithms, which have been studied in great detail for single-destination problems, but not previously for multiple destinations.

4. THE FIRST-COME FIRST-SERVE ALGORITHM WITH ONE DESTINATION

Splitting algorithms for contention resolution, which are inherently stable algorithms, provide a higher throughput than that which can be achieved using stabilized slotted Aloha. Splitting can be implemented in several ways, including coin toss, node ID, or time of arrival. Recently, the idea of splitting based on residual energy has been introduced [4]. In this paper, we consider the use of First-Come First-Serve (FCFS) schemes. Splitting algorithms are based on the immediate availability of ternary feedback information (success, collision, or empty slot), transmitted by the destination, which describes the outcome of the slot.

We first consider the FCFS splitting algorithm for a single destination, which is discussed in detail in [3]. Under this scheme, a collision-resolution epoch is initiated when a collision occurs. All packets that arrive within a specified "allocation interval" are transmitted in the first slot of this epoch. If there is a collision, this window is shortened, and the process is repeated.

The achievable throughput can be increased by means of two improvements [3]. The first addresses the case of a collision that is followed immediately by an idle slot. Straightforward application of the FCFS algorithm guarantees that a collision will occur in the next slot. To avoid such wasting of a slot, the algorithm acts as if the collision had already occurred, and shortens the allocation interval accordingly. The second improvement (which has a somewhat smaller impact on achievable throughput) incorporates the right (i.e., later) half of the allocation interval involved in the original collision into subsequent collision resolution periods. The combined effect of using both improvements provides a maximum stable throughput of 0.4878.

It is noted in [3] (as originally observed in [5]), that use of the first improvement makes the algorithm vulnerable to errors in the feedback process. Specifically, consider the case in which an idle slot is erroneously interpreted as a collision. Since it is believed that a collision has occurred, the allocation interval is then split. However, since there are actually no packets in the allocation interval, the result is an empty slot. Use of the first improvement then results in further reduction of the empty allocation interval, a process that continues indefinitely, resulting in deadlock. (By contrast, it was observed in [5] that, when the first improvement is not used, splitting algorithms are extremely insensitive to feedback errors.) In applications where such errors are rare, it may be appropriate to simply suspend the splitting of the interval after observing several such idle slots. However, it may be best not to use the first improvement at all when such errors are frequent. The case of multiple destinations creates a situation that is similar to that of frequent feedback errors, as we discuss in the next section.

[2] This assumption deviates slightly from the true classic Aloha scheme; usually the retransmission probability is chosen to either stabilize or control Aloha's behavior. However, here, the attention to Aloha is solely for establishing benchmark behavior and is not of interest per se. Thus the assumed constant retransmission probability represents a minor inconsequential deviation.

5. THE FIRST-COME FIRST-SERVE ALGORITHM WITH TWO DESTINATIONS

We first consider the most straightforward implementation of the FCFS algorithm with two destinations, in which the first improvement is not used. For example, D1's neighbors are all the nodes in Group 1 and Group 3. D1 hears all of its neighbors' transmissions (even though some of the Group 3 transmissions are intended for D2). As in the single-destination case, D1's feedback for a slot in which none of its neighbors transmits is 0 (for an empty slot). If exactly one of its neighbors transmits, we must distinguish between two cases; the feedback is 1 (for a success) if the packet was, in fact, intended for D1; it is 0 if the packet is a Group 3 packet that was intended for D2. Whenever more than one of D1's neighbors transmits (regardless of their intended destinations), the feedback is *e* (for collision). In this simple implementation, nodes in Group 3 make use of the feedback transmitted by only their intended destination, even though they receive feedback from both destinations (which can potentially be used to improve performance). There is no coordination of the collision-resolution processes at the two destinations. Their allocation intervals are determined independently, as are the beginning and end points of the collision resolution intervals.

The difference between this problem and that of a single destination results from the interference (collisions) caused by packets in Group 3 that are intended for D2. We noted above that splitting algorithms (for a single destination) are extremely insensitive to feedback errors when the first improvement is not used. We have observed in our simulations of the two-destination problem that the straightforward implementation of the FCFS algorithm functions correctly (albeit at reduced throughput levels) despite the introduction of collisions with packets intended for the other destination. Before discussing the throughput levels that are achievable, we address the deadlock that can arise if the first improvement is used in the two-destination problem, which is similar to that which results from feedback errors, as discussed above. We then discuss a simple technique that eliminates the deadlock mechanism.

Figure 2 illustrates an example that leads to deadlock if the first improvement is used. Suppose that we start with an empty system, and that three packets (A, B, and C) are generated during time slot 0. In this case, to simplify the figure, we implicitly assume that the initial value of the allocation interval for both D1 and D2 is equal to a slot length, although there is in general no direct relationship between allocation interval (a continuous variable) and slot length (which is fixed). What is important here is that A and B arrive during the first half of the allocation interval corresponding to D1 and that C arrives during the second half of the allocation interval corresponding to D2 (where in both cases, the initial allocation period for both D1 and D2 coincides with slot 0).

In our example, both A and B are from Group 3 and are intended for D1, while C is from Group 2 and is intended for D2. All three packets attempt transmission in slot 1, and both D1 and D2 experience collisions (note that all three transmissions reach D2). As a result of these collisions, both D1 and D2 will cut their allocation intervals in half; therefore, packets that arrived during the first half of slot 0 (namely, A and B, but not C) will transmit during slot 2. Again, a collision will be observed at both D1 and D2. D1's allocation interval will then be halved again, which may permit the successful transmission of A in slot 3 and B in slot 4 (depending on their actual arrival times within the first half of slot 0). However, D2's allocation interval will also be halved, with the first half (quarter of the original allocation interval) being assigned to slot 3. Upon finding this interval to be empty, and thus assuming that a collision is guaranteed in the next slot, the second half will be split (the first half of which will again be found to be empty), and so on. Therefore, the allocation interval for D2 will never extend beyond the third quarter of slot 0. Consequently, transmission of packet C, and thereafter all packets in Group 2, will be suspended indefinitely.

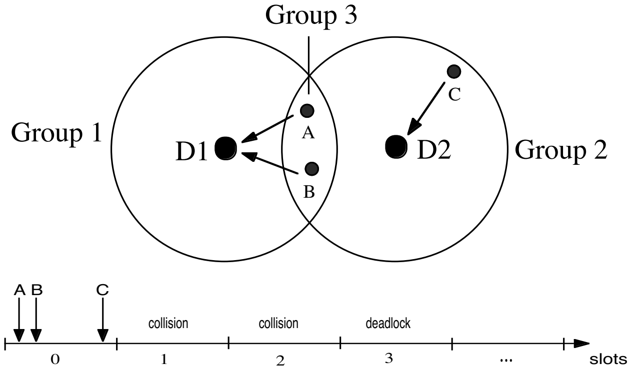

Fig. 2 — Example of deadlock when the first improvement is used in a two-destination network.

This type of deadlock is not a rare event, and the high likelihood of its occurrence, even in networks with no feedback errors, suggests that it is not advisable to use the first improvement in multiple-destination scenarios, unless a means is developed to eliminate the possibility of such deadlocks.

One possible approach to eliminate the complications created by the two destinations is to impose a time-division structure that creates a time orthogonality among groups of nodes that interfere with each others' feedback process. For example, consider Fig. 1. The nodes in Group 1 and Group 2 can transmit simultaneously without any interference between the feedback of the two groups. The nodes in Group 3 would interfere with those in both Group 1 and Group 2; moreover, Group 3 nodes transmitting to D1 would interfere with those that transmit to D2. Let us assume that our goal is to determine an appropriate slot assignment that maximizes the throughput that can be achieved.[3] This situation suggests that three classes of slots be allocated proportionately to the packet rate in each class, namely:

fraction of slots allocated to Group 1 and Group 2 nodes:

$$\frac{1-f}{1+f}$$

[3] A variety of similar problems can also be considered. For example, if the fraction of total traffic associated with nodes in each Group is specified, it would be appropriate to determine the slot allocation that provides the maximum achievable throughput, as well as the throughput that can actually be achieved.

fraction of slots allocated to Group 3 nodes transmitting to D1:

$$\frac{f}{1+f}$$

fraction of the slots allocated to Group 3 nodes transmitting to D2:

$$\frac{f}{1+f}$$

Such a time-division structure would eliminate deadlock situations such as the one described above, and would therefore permit the use of the first improvement (assuming that feedback errors caused by noise are negligible).

Let us consider the problem of determining the maximum throughput per destination that can be supported in the network of Fig. 1. We assume a symmetrical system with an infinite number of nodes, in which the combined arrival rate of Group 1 nodes is equal to that of Group 2 nodes. We also specify the fraction f of the overall arrival rate of nodes in Group 3, half of which is destined for D1 and half for D2. It can be shown that the maximum achievable throughput per destination is [6]

$$S_{norm} = \frac{S}{1+f}$$

where S is the maximum throughput achievable by the FCFS splitting algorithm (i.e., 0.4878 if both improvements are used). Another slot allocation can yield throughput that is bounded above by $(1-f/2)S$ [6].

The upper curve in Fig. 3 shows the maximum throughput per destination that can be achieved by allocating the time slots according to the fractions discussed above, as a function of f. The lower curve represents the use of the FCFS splitting algorithm independently at the two nodes, where only the second improvement is used (to avoid the deadlock situation discussed above). We observe that the orthogonal time division of transmissions can provide better performance, largely because it permits use of the first improvement. We also observe that reasonably good performance can be achieved by a very simple implementation, in which the FCFS algorithm is implemented without any effort to coordinate the collision-resolution process at the two destinations.

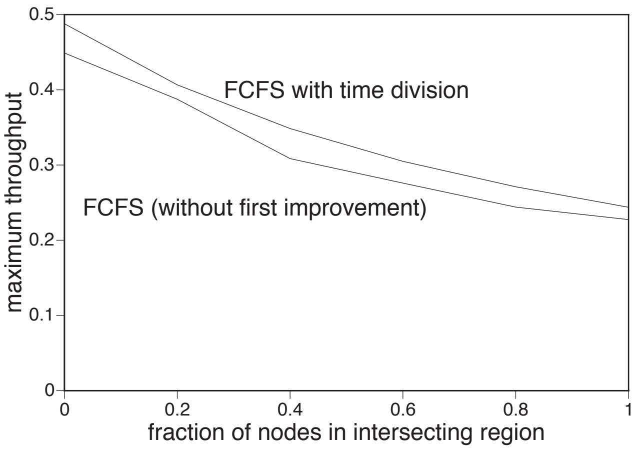

Fig. 3 — Throughput per destination for two-destination collision resolution, as a function of the fraction of nodes in the intersecting region.

6. SUMMARY AND CONCLUSIONS

In this paper, we have addressed the MAC problem for multiple destinations, and have described the issues associated with using time-of-arrival-based splitting algorithms in such environments. We have demonstrated the susceptibility of the FCFS splitting algorithm to deadlock when a well-known throughput-increasing improvement is incorporated. However, we have also demonstrated, via simulation, that a straightforward implementation (without this improvement) is not subject to this deadlock. Furthermore, we have shown that the benefits of the improvement can be exploited if time division is used to partition the groups of users so that they do not interfere with each other.

Our future work will address techniques to improve the logic of the collision resolution algorithm, e.g., the nodes in Group 3 may be able to exploit the feedback information from both nodes, rather than only the node to which they are transmitting. We will also address the case of multiple destinations. The extension of our time-division approach to multiple destinations is related to the link-activation problem (see, e.g., [7], [8], [9]), in which non-interfering nodes (i.e., nodes that are sufficiently far from each other) are permitted to transmit simultaneously.

REFERENCES

[1] A. J. Goldsmith and S. B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks," *IEEE Wireless Communications,* pp. 2-22, August 2002.

[2] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?," *IEEE Communications Magazine,* **vol. 39**, pp. 130-137, June 2001.

[3] D. Bertsekas and R. Gallager, *Data Networks,* Englewood Cliffs: Prentice Hall, 1992.

[4] Y. E. Sagduyu and A. Ephremides, "Energy-Efficient Collision Resolution in Wireless Ad-Hoc Networks," to appear in *Proc. IEEE Infocom 2003,* 2003.

[5] J. L. Massey, "Collision-Resolution Algorithms and Random-Access Communications," in *Multi-User Communication Systems,* Edited by G. Longo, CISM Courses and Lectures No. 265, New York: Springer-Verlag, pp. 73-137, 1981.

[6] G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Contention-Resolution Algorithms for Multiple Destinations in Wireless Networks," Submitted to *2003 Conference on Information Sciences and Systems,* Johns Hopkins University, March 2003.

[7] D. J. Baker, A. Ephremides, and J. A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," *IEEE J. SAC,* **SAC-2**-No. 1, pp. 226-237, January 1984.

[8] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proc. IEEE,* **75**-No. 1, pp. 56-73, January 1987.

[9] A. Ephremides and T. V. Truong, "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Trans. Commun.,* **38**, pp. 456 - 460, April 1990.