MAC Design for Analog Network Coding*

Majid Khabbazian
University of Winnipeg
Canada
m.khabbazian@uwinnipeg.ca

Fabian Kuhn
University of Lugano (USI)
Switzerland
fabian.kuhn@usi.ch

Nancy Lynch
MIT Computer Science and Artificial Intelligence Lab
lynch@csail.mit.edu

Muriel Médard
MIT Research Laboratory of Electronics
medard@mit.edu

Ali ParandehGheibi
MIT Research Laboratory of Electronics
parandeh@mit.edu

ABSTRACT

Most medium access control (MAC) mechanisms discard collided packets and consider interference harmful. Recent work on Analog Network Coding (ANC) suggests a different approach, in which multiple interfering transmissions are strategically scheduled. Receiving nodes collect the results of collisions and then use a decoding process, such as ZigZag decoding, to extract the packets involved in the collisions.

In this paper, we present an algebraic representation of collisions and describe a general approach to recovering collisions using ANC. To study the effects of using ANC on the performance of MAC layers, we develop an ANC-based MAC algorithm, *CMAC*, and analyze its performance in terms of probabilistic latency guarantees for local packet delivery. Specifically, we prove that *CMAC* implements an *abstract MAC layer* service, as defined in [14, 13]. This study shows that ANC can significantly improve the performance of the abstract MAC layer service compared to conventional probabilistic transmission approaches.

We illustrate how this improvement in the MAC layer can translate into faster higher-level algorithms, by analyzing the time complexity of a multi-message network-wide broadcast algorithm that uses *CMAC*.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*;
C.2.2 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

*Based upon work under subcontract #18870740-37362-C by Stanford U. and supported by DARPA. Further, supported in part by AFOSR grant FA9550-08-1-0159, NSF grants CCF-0726514 and CCF-0937274, and by NSERC.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
FOMC'11, June 9, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0779-6/11/06 ...$10.00.

General Terms

Algorithms, Theory

Keywords

analog network coding, MAC layer, multi-message broadcast, wireless network algorithms

1. INTRODUCTION

The nature of wireless networks is intrinsically different from that of wired networks because the wireless medium is shared among many transmitters. The conventional approach to the Medium Access Control (MAC) problem is to use contention-based protocols in which multiple transmitters simultaneously attempt to access the wireless medium, operating under rules that provide enough opportunities for all nodes to transmit. Well-known examples of such protocols in packet radio networks are ALOHA [2], MACAW [4], and CSMA/CA [3].

In contention-based protocols it is possible that two or more nodes transmit their packets simultaneously, which can result in a *collision* at a receiver. The colliding packets are generally considered to be lost. Therefore, these protocols strive to avoid simultaneous transmissions by nearby nodes. Recently, Gollakota and Katabi [9] showed how one might recover collided packets in an 802.11 system using *ZigZag decoding*, if there are relatively few colliding packets and enough transmissions involving these packets. Their scheme requires the network to operate at a signal-to-noise ratio (SNR) that is sufficiently high that noise can be neglected and per-symbol detection can be assumed to be error-free in the absence of a collision on that symbol. In fact, they suggest that each collision can be treated as a linear equation over the packets involved. Therefore, packets are recoverable if the resulting system of linear equations has a unique solution. This gives rise to the possibility of designing MAC protocols that exploit Analog Network Coding (ANC) [10] to increase network capacity. In such MAC protocols, unlike conventional protocols, interference is not considered harmful. In fact, such protocols strategically schedule simultaneous transmissions in order to increase network capacity. Note that, as in digital network coding, packets are mixed together in ANC. However, in digital network coding, the sender mixes the contents of packets before transmission whereas in ANC the wireless channel naturally mixes the packets by adding the signals.

In this paper, we present a new MAC protocol, *CMAC*, which exploits Analog Network Coding, and prove that it provides strong performance guarantees, in terms of probabilistic latency bounds for local packet delivery. Specifically, we prove that *CMAC* implements the formal *probabilistic abstract MAC layer specification* introduced by Khabbazian *et al.* in [14, 13] This specification assumes that packets arrive at the MAC layer at arbitrary times, but with at most one packet active at each node at any point. This specification provides probabilistic upper bounds on the time for a packet to be delivered to any neighboring node (the *receive delay bound*), and on the total amount of time for a sender to receive an acknowledgment of successful delivery to all neighbors (the *acknowledgment delay bound*). It also provides a bound on the amount of time for a receiver to receive *some* packet from among those currently being transmitted by neighboring senders (the *progress bound*).

Note that this specification describes a MAC layer that provides guarantees for *local broadcast*, not local unicast; this captures the fundamental broadcast capability of wireless networks. The receive and acknowledgment delay bounds should be self-explanatory. The progress bound captures the fact that, in many situations, the time required for a receiver to receive a packet from *some neighbor* is considerably shorter than the time needed to obtain a packet from a *specific neighbor*. This difference turns out to be significant for the performance of some higher-level protocols, such as certain network-wide broadcast protocols, where any new information is useful for advancing the protocol.

In Section 6, we prove that *CMAC* implements the probabilistic MAC layer with receive, acknowledgment, and progress bounds all of the form $\mathcal{O}\left(\Delta + \frac{\Delta}{c}\log\frac{\Delta}{\epsilon}\right)$, where Δ is the maximum node degree and c is the maximum number of packets for which a collision can be decoded. In particular, if c is $\Omega(\Delta)$, then using ANC, a node can deliver a packet to all of its neighboring nodes (and receive a packet from any given neighboring node) in time $\mathcal{O}\left(\Delta + \log\frac{1}{\epsilon}\right)$.

These latency bounds for *CMAC* allow us to compare *CMAC* with more conventional MAC protocols. For example, Khabbazian *et al.* [14, 13] described a conventional probabilistic transmission MAC protocol, *DMAC*, which uses exponential decay. They showed that *DMAC* has receive and acknowledgment delay bounds of the form $\mathcal{O}\left(\Delta\log\left(\frac{1}{\epsilon}\right)\log\Delta\right)$, which are larger than those of *CMAC*, but has a smaller progress bound, of the form $\mathcal{O}(\log\Delta)$. In Section 7, we present another MAC protocol, *DCMAC*, which improves the progress bound of *CMAC* to $\mathcal{O}(\log\Delta)$ without increasing the receive and acknowledgment bounds. *DCMAC* achieves these bounds by interleaving *CMAC* and *DMAC*.

Showing that *CMAC* implements the probabilistic abstract MAC layer yields another important benefit: it allows us to analyze the effect of using ANC on the time complexity of higher-level algorithms such as in particular network-wide broadcast algorithms. As an example, in Section 8, we combine our analysis of *CMAC* with an analysis of a high-level multiple-message global broadcast protocol over the probabilistic MAC layer from [14, 13], thus obtaining time bounds for multi-message broadcast over the basic network. Our results show that the time complexity of multi-message broadcast can be significantly improved using ANC, as compared to conventional probabilistic transmission.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. Section 3 presents our network assumptions, including the key facts about Analog Network Coding. Section 4 describes and analyzes a simple one-hop ANC algorithm. Section 5 describes the probabilistic abstract MAC layer, as defined in [14, 13]. Sections 6 and 7 present our two algorithms, *CMAC* and *DCMAC*. Section 8 describes a simple network-wide broadcast algorithm that uses *DCMAC* and analyzes its time complexity. Section 9 concludes. Complete details of this work appear in our Technical Report [15].

Notation: Let $\mathbb{C}$ denote the set of complex numbers. We use log to denote the base two logarithm and ln the natural logarithm. For any positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$.

2. RELATED WORK

Analog Network Coding was first presented in [10]. For high SNR regimes, the asymptotic optimality of ANC was recently shown in [19, 20]. Its asymptotic optimality in terms of rate-diversity tradeoff was established in [6, 5]. The use of ANC, as a generalization of ZigZag, in possible combination with digital network coding, in order to increase the throughput of packetized multiple-access systems, has been considered in [22]. In that work, an algebraic model for ANC was derived, which explicitly takes into account symbols, digital network coding, modulation, and channel effects such as attenuation, delay and additive combination of signals. We use that model in this paper to model the algebraic interaction among nodes. The use of ZigZag without additional digital network coding has recently been considered by [23] to improve congestion control and maximize aggregate utility of the users. That approach does not describe how to implement a MAC specification, as we do in this paper. In another related paper, Zhang, *et al.* [24] describe an algorithm for physical-layer network coding. However, the proposed algorithm assumes symbol-level synchronization, carrier-frequency synchronization, and carrier-phase synchronization. In contrast, the algorithms in [10] and [9] make no such synchronization assumptions.

The first abstract MAC layer specification was defined by Kuhn, Lynch, and Newport [16, 17, 18]. This basic layer provides worst-case latency guarantees for packet receipt and acknowledgment. These papers also present and analyze greedy network-wide broadcast algorithms over the basic MAC layer. Khabbazian, *et al.* [14, 13] continued this work by developing the probabilistic version of the MAC layer specification that is used in this paper, presenting probabilistic transmission algorithms to implement both layers, analyzing network-wide broadcast algorithms over both layers, and showing how to combine the high-level and low-level results systematically to obtain performance results for network-wide broadcast over a collision-prone radio network. Other work using abstract MAC layers includes algorithms for Neighbor Discovery [7, 8].

3. THE NETWORK MODEL

Fix a static undirected graph, $G = (V, E)$. Let $n = |V|$ be the number of vertices, Δ the maximum degree, and D the diameter, i.e., the maximum distance (in terms of number of hops) between any two vertices in G. We assume that n active nodes reside at the n vertices of G. Nodes have unique identifiers. We assume that the nodes have local knowledge of the graph; in particular, they know Δ and know the identifiers of their neighbors in G.

We assume a slotted system, with slots of duration $t_{slot} = 1$. When a node transmits in some slot, its message *reaches* all G-neighboring nodes, and no other nodes. Thus, each node j, in each slot, is reached by some collection of packets from its transmitting neighbors. What j actually *receives* is defined as follows: (a) If j transmits, then it receives silence, denoted by $\perp$. Thus, a node cannot receive a packet while it is transmitting. (b) If j does not transmit and is reached by no packets, then it receives silence. (c) If j does not transmit and is reached by exactly one packet from another node, then it receives that packet. (d) If j does not transmit and is reached by two or more packets, then it receives a collision. We assume that each node stores all the received packets and collisions, and uses analog network coding (such as ZigZag decoding) to decode the collided packets.

A packet is essentially a vector of N symbols over a finite field $\mathbb{F}_q$, where q is a power of two. We represent a packet as a polynomial with coefficients being the symbols of $\mathbb{F}_q$ that form the packet. The mapping from the packet to the corresponding physical signal is a result of modulation. For a system such as ZigZag, which performs per-symbol detection, no channel coding precedes the modulation. For more general ANC, however, there may also be a channel code, requiring the use of interference cancellation over the entire packet, rather than symbol-wise operations as in ZigZag. For the sake of simplicity, we discuss here the case where no channel code is added, although our discussion can be extended to the case where we have channel coding (because the effect of the noise is not entirely negligible on a symbol-by-symbol bases). We abstract the modulation to be a one-to-one map M from symbols over $\mathbb{F}_q$ to the complex number field

$$M : \mathbb{F}_q \to \mathbb{C}.$$

Using the model of [22], an equivalent representation of the collisions of w packets at receiver j in l time slots is given by

$$\begin{pmatrix} C_j^{s_1}(k) \\ C_j^{s_2}(k) \\ \vdots \\ C_j^{s_l}(k) \end{pmatrix} = \begin{pmatrix} A_{i_1}^{s_1} & \dots & A_{i_w}^{s_1} \\ A_{i_1}^{s_2} & \dots & A_{i_w}^{s_2} \\ \vdots & & \vdots \\ A_{i_1}^{s_l} & \dots & A_{i_w}^{s_l} \end{pmatrix} \begin{pmatrix} S_{i_1}(k) \\ S_{i_2}(k) \\ \vdots \\ S_{i_w}(k) \end{pmatrix},$$

$k = 1, \dots, N$, where $C_j^s \in \mathbb{C}^N$ represents the collision in time slot s, $S_i \in \mathbb{C}^N$ is the signal (packet) transmitted by sender i, and $A_i^s \in \mathbb{C}$ are random variables corresponding to the combination of the modulation and channel propagation effects, as well as the transmission decision of sender i at time slot s.

Note that ZigZag relies on there being non-zero time shifts among colliding packets, whereas general ANC does not. The process of decoding by inverting this matrix is more general than the ZigZag procedure of [9]. For example, consider the case where two different nodes collide twice in two different time slots. If the offsets between two packets in the two time slots are exactly the same, the ZigZag decoding process fails. However, the transfer matrix A may still be full-rank because of the change in the channel gains over time, and hence, we may decode the packets by Proposition 1. The decoding process results in the signals corresponding to the original packets. The signals then have to be demodulated to obtain the original data. This algebraic representation formalizes the intuition introduced in [9] that every collision is like a linear equation in the original packets. Let

$$A = \begin{pmatrix} A_{i_1}^{s_1} & A_{i_2}^{s_1} & \dots & A_{i_w}^{s_1} \\ A_{i_1}^{s_2} & A_{i_2}^{s_2} & \dots & A_{i_w}^{s_2} \\ \vdots & \vdots & & \vdots \\ A_{i_1}^{s_l} & A_{i_2}^{s_l} & \dots & A_{i_w}^{s_l} \end{pmatrix}.$$

PROPOSITION 1. *Let P, $|P| = w$, be a set of packets. Consider a node j. Let S, $|S| = l$, be a set of slots such that in every slot in S, node j receives a packet in P or a collision involving packets in P. The received packets/collisions can be represented by a system of linear equations of the form $C(k) = A \times S(k)$, where $k = 1, 2, \dots, N$ and A is an $l \times w$ transfer matrix. Given this representation, if the transfer matrix A has rank w (i.e., full rank) over the field $\mathbb{C}$, then it is possible to decode all packets in P.*

Proposition 1 follows from techniques described in [9, 22]. Note that the coding coefficients are not chosen by the transmitters of the messages. They are mainly dictated by the channel conditions and can be estimated through "physical layer" techniques. For example, a *correlation* technique is described in [9] to accurately estimate channel conditions. Several other practical issues such as sampling offset, frequency offset and phase tracking are thoroughly discussed in [10] and [9]. Nevertheless, it is fair to say that the assumptions in our model are well justified through practical implementation of the ZigZag decoding algorithm on a software radio platform.

One limitation of analog network coding and collision recovery techniques is that they require sufficiently high SNR so that the effect of noise can be safely removed. In practice, since the received signals are noisy, as the number of packets involved in a collision increases, it becomes less likely that the collision can be used for decoding. Hence, towards a more realistic setup, we assume that any collision involving more than c (a fixed parameter) packets is not useful and will be discarded. The parameter c *indirectly* captures the effect of noise as well as the physical layer detection algorithm, without requiring us to change the model for different physical layer techniques. For instance, we may use the SigSag decoding algorithm of Tehrani *el al.* [1], an improvement on the ZigZag decoding algorithm, to significantly reduce the effect of noise. Hence, the parameter c for SigSag would be larger than that for ZigZag decoding. Throughout the paper, we assume that $4 \le c \le \Delta$. Note that, based on our network assumptions, at most Δ packets may be involved in a collision. Thus, a decoder with parameter $c = \Delta$ is as powerful as one with parameter $c > \Delta$.

4. BASIC CODING STRATEGY

In this section, we describe and analyze a simple single-hop ANC-based contention-resolution protocol, which we call simply *Coding*. We use this protocol in our *CMAC* algorithm, in Section 6.

4.1 Probability that a Matrix is Full-Rank

The heart of the analysis of *Coding* is a mathematical lemma, Lemma 2, which expresses a lower bound on the probability that a matrix B generated randomly from an arbitrary matrix A has full rank. In Section 4.2, we use such random matrices to model the transmission behavior

of the neighbors of a particular node k, where entry $B_{i,j}$ corresponds to the transmission behavior of node j in slot i. The conclusion of Lemma 2 is used there to show that, assuming that the algorithm executes for enough slots, with high probability, node k receives enough information to recover a set of packets.

> **Construction of a random matrix:** Let ℓ and w be positive integers. Let A be an arbitrary matrix of size $\ell \times w$, with elements in $\mathbb{C} - \{0\}$. Let p be a real number, $0 \le p \le 1$. We construct a random $\ell \times w$ matrix B from A according to the following two-step procedure: Start with $B = A$. First, for each i independently, keep row i of B unchanged with probability p and set it to be identically 0 with probability $1 - p$. Second, for each non-zero row i and each column j, independently, keep $B_{i,j}$ unchanged with probability p and set it to 0 with probability $1 - p$.

LEMMA 2. *Let ℓ and w be positive integers. Let A be an arbitrary matrix of size $\ell \times w$, with elements in $\mathbb{C} - \{0\}$. Further, let p and ϵ be real numbers, with $0 < p \le \frac{1}{2}$ and $0 < \epsilon < 1$. Let c be a positive integer with $c \ge 4$ and $wp \le \frac{c}{2}$. Suppose that*

$$\ell \ge \left\lceil \frac{\frac{14e}{e-1}}{1-p} \left(w + \frac{\ln(w+1) + 2\ln(1/\epsilon)}{p} \right) \right\rceil.$$

Let B be a random matrix constructed from A as described above. Then, with probability at least $1 - \epsilon$, B has at least w independent rows, each containing at most c non-zero elements.

PROOF (SKETCH). Instead of fixing the number of rows of B, assume that we construct B by adding rows until there are w rows with at most c non-zero entries that span $\mathbb{R}^w$. The lemma then follows by showing that with probability at least $1 - \epsilon$, the resulting matrix has at most ℓ rows.

For each $d \in [w]$, we define random variables X_d and Y_d. Let X_d be the smallest integer such that the sub-matrix spanned by the rows with at most c non-zero entries among the first X_d rows of B has rank d. Further, we define $X_0 = 0$ and $Y_d = X_d - X_{d-1}$. Note that to prove the lemma, we need to show that $\Pr(X_w > \ell) < \epsilon$.

In the following, we refer to non-zeroed rows of B as the rows that are not set to 0 at the end of the construction of B. The non-zeroed rows of B are random vectors in $\mathbb{C}^w$, where each coordinate is non-zero independently with probability p. Assume that we are given $d - 1$ linearly independent vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d-1}$ for some integer $d \ge 1$. The core of the proof is to lower bound the probability that a new non-zeroed row of B has at most c non-zero entries and is linearly independent to the given $d - 1$ vectors. For simplicity, assume that the vectors $\boldsymbol{x}_i$ are vectors from the standard basis of $\mathbb{C}^w$. Hence, each of them is non-zero in exactly one coordinate and an additional vector is linearly independent iff it is non-zero in at least one coordinate in which none of the $d - 1$ vectors $\boldsymbol{x}_i$ is non-zero. The probability that all these $w - d + 1$ coordinates are 0 in a random non-zeroed row is $(1 - p)^{w-d+1}$ since all coordinates are set to something non-zero with probability p. The parameter c is chosen such that the probability that a non-zeroed row has at most c non-zero entries is lower bounded by some constant q. It can be shown that the probability that a non-zeroed row has at most c non-zero entries and is linearly independent of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d-1}$ is at least

$$p_d = \frac{1}{7} \cdot \left(1 - (1-p)^{w-d+1}\right).$$

Further it can be shown that the same bound holds if the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d-1}$ are arbitrary vectors in $\mathbb{C}^w$. Therefore for each $d \ge 0$ and row $i > X_{d-1}$, if row i is not set to 0 in the final step of the construction of B, the probability that row i contains at most c non-zero entries and is independent of the first X_{d-1} rows is at least p_d. Thus, the random variables Y_d are dominated by independent geometric random variables Z_d with parameter $(1 - p)p_d$. Using a Chernoff bound, it can be shown that

$$\Pr(X_d \le \ell) \le \Pr\left(\sum_{i=1}^{d} Z_d \le \ell\right) \le 1 - \epsilon.$$

This completes the proof sketch. Details of the proof appear in [15]. □

4.2 The Coding Algorithm

Now we describe and analyze the *Coding* protocol. Let c be the threshold parameter defined for ANC (in Section 3). Let $\rho = \frac{c}{2\Delta}$. Note that $\rho \le \frac{1}{2}$, because $c \le \Delta$.

DEFINITION 1 ($\mathcal{R}_\epsilon$, WHERE ϵ IS A REAL, $0 < \epsilon < 1$).

$$\mathcal{R}_\epsilon = \left\lceil \frac{\frac{14e}{e-1}}{1-\rho} \left(\Delta + \frac{\ln(\Delta+1) + 2\ln(1/\epsilon)}{\rho} \right) \right\rceil$$

LEMMA 3. $\mathcal{R}_\epsilon = \mathcal{O}\left(\Delta + \frac{\Delta}{c} \log \frac{\Delta}{\epsilon}\right)$.

The *Coding* algorithm has a single explicit parameter, ϵ. It also uses c as an implicit parameter.

> **Coding**(ϵ), where ϵ is a real, $0 < \epsilon < 1$: Assume I is a set of nodes, $1 \le |I| \le \Delta$, and j is a distinguished node adjacent to all nodes in I. All the nodes in I participate in the algorithm, and j may or may not participate. Each participating node i has a packet m_i, assumed fixed for the entire algorithm.
> The algorithm runs for exactly $\mathcal{R}_\epsilon$ slots. Every participating node participates in all slots, with no node starting or stopping participation part-way through the algorithm. At every slot, each participating node i transmits packet m_i with probability $\rho = \frac{c}{2\Delta}$.

LEMMA 4. *In Coding(ϵ), with probability at least $1 - \epsilon$, node j receives all packets m_i, $i \in I$, by the end of the algorithm.*

PROOF. The probability that node j receives all packets if it participates in the algorithm is at most equal to the probability that j receives all packets if it does not participate (if the node participates, it can only receive in time slots where it does not transmit, otherwise, it can receive in all time slots). So we assume without loss of generality that j participates.

We construct a random matrix B of size $\mathcal{R}_\epsilon \times |I|$, in $\mathcal{R}_\epsilon$ steps. In step r, $1 \le r \le \mathcal{R}_\epsilon$, we define row r, as follows. If j transmits in slot r, then row r is identically 0. If j does not transmit in slot r, then write $I = \{i_s | 1 \le s \le |I|\}$. For every s, $1 \le s \le |I|$, let $B_{r,s} = 0$ if node i_s does not transmit

in slot r; otherwise let $B_{r,s}$ be a non-zero complex number (which corresponds to the channel gain from node i_s to node j and the offset of i_s's packet).

Now we apply Lemma 2, with ℓ in the statement of that lemma equal to $\mathcal{R}_\epsilon$, w in the lemma equal to $|I|$, and $p = \rho$. Since

$$|I|\rho = |I|\frac{c}{2\Delta} \leq \frac{c}{2}$$

and

$$\ell = \mathcal{R}_\epsilon = \left\lceil \frac{\frac{14e}{e-1}}{1-\rho}\left(\Delta + \frac{\ln(\Delta+1) + 2\ln(1/\epsilon)}{\rho}\right)\right\rceil,$$

the conditions required by Lemma 2 hold. Consequently, by Lemma 2, with probability at least $1-\epsilon$, B has a set S of $|I|$ independent rows, each containing at most c non-zero elements. Any row in S corresponds to a collision that j receives during the execution of $Coding(\epsilon)$. Consequently, by Proposition 1, j can decode all the packets m_i, $i \in I$, by the end of the algorithm, with probability at least $1-\epsilon$. □

5. PROBABILISTIC MAC LAYER SPECIFICATION

Now we are ready to consider MAC layers. Before presenting our ANC-based MAC algorithm *CMAC*, we give a formal specification for MAC layer requirements. For this, we use the *probabilistic abstract MAC layer specification* from [14, 13]. This specification describes MAC-layer behavior in a multi-hop network. It assumes that packets arrive from the environment (a higher-level protocol) nondeterministically, at arbitrary times, and not according to any predetermined probability distribution. We assume that at most one packet is active at a time at each node, that is, the environment waits for a node to complete its processing of one packet before it provides that node with a new packet. We do not assume that every node always has an active packet. Our service provides guarantees for local broadcast rather than local unicast, reflecting the fundamental broadcast capability of wireless networks.

According to our specification, a MAC layer provides an external interface by which it accepts packets from its environment via $bcast(m)$ input events and delivers packets to neighboring nodes via $rcv(m)$ output events. It also provides acknowledgments to senders indicating that their packets have been successfully delivered to all neighbors, via $ack(m)$ output events. Finally, it accepts requests from the environment to abort current broadcasts, via $abort(m)$ input events.

The specification is implicitly parameterized by three positive reals, f_{rcv}, f_{ack}, and f_{prog}. These bound delays for a specific packet to arrive at a particular receiver, for an acknowledgment to be returned to a sender, and for *some* packet from among many competing packets to arrive at a receiver. The specification also has corresponding parameters ϵ_{prog}, ϵ_{rcv}, and ϵ_{ack}, which represent bounds on the probabilities that the delay bounds are not attained. Finally, it has a parameter t_{abort}, which bounds the amount of time after a sender aborts a sending attempt when the packet could still arrive at some receiver.

We model a MAC layer formally as a *Probabilistic Timed I/O Automaton* (PTIOA), as defined by Mitra [21]. A MAC layer PTIOA *Mac* is composed with an environment PTIOA *Env* and a network PTIOA *Net*. This composition, written as $Mac \| Env \| Net$, is itself a PTIOA, and yields a unique probability distribution on executions (once nondeterminism is resolved using various scheduling mechanisms, see [21]).

To satisfy our specification, a MAC layer *Mac* must guarantee several conditions, when composed with any *Env* and with *Net*. To define these requirements, we assume some simple constraints on *Env*, namely, we consider executions α of $Mac \| Env \| Net$ that are *well-formed*, in the sense that: (a) they contain at most one *bcast* event for each m (i.e., all packets are unique), (b) any $abort(m)_i$[1] event is preceded by a $bcast(m)_i$ but not by an $ack(m)_i$ or another $abort(m)_i$, and (c) any two $bcast_i$ events have an intervening ack_i or $abort_i$ (i.e., each node handles packets one at a time).

The specification says that the *Mac* automaton must guarantee the following conditions, for any well-formed execution α of $Mac \| Env \| Net$. There exists a *cause* function that maps every $rcv(m)_j$ event in α to a preceding $bcast(m)_i$ event, $i \neq j$, and that maps each $ack(m)_i$ and $abort(m)_i$ to a preceding $bcast(m)_i$. The *cause* function must satisfy:

- **Receive restrictions:** If $bcast(m)_i$ event π causes $rcv(m)_j$ event π', then (a) *Proximity:* $(i,j) \in E$. (b) *No duplicate receives:* No other $rcv(m)_j$ caused by π precedes π'. (c) *No receives after acks:* No $ack(m)_i$ caused by π precedes π'. (d) *Limited receives after aborts:* π' occurs no more than t_{abort} time after an *abort* caused by π.

- **Acknowledgment restrictions:** If $bcast(m)_i$ event π causes $ack(m)_i$ event π', then (a) *No duplicate acks:* No other $ack(m)_i$ caused by π precedes π'. (b) *No acks after aborts:* No $abort(m)_i$ caused by π precedes π.

In addition, the *Mac* automaton must guarantee three probabilistic upper bounds on packet delays—a *receive delay bound*, an *acknowledgment delay bound*, and a *progress bound*. Thus, if π is a *bcast* event in a closed execution β[2], then we say that π is *active at the end of* β provided that π is not terminated with an *ack* or *abort* in β. The probabilistic MAC layer guarantees the following probabilistic bounds. Here, the notation Pr_β refers to the conditional distribution on executions that extend β. Assume $i, j \in V$, and t is a nonnegative real.

- **Receive delay bound:** Let β be a closed execution that ends with a $bcast(m)_i$ at time t. Let j be a neighbor of i. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m)_i$ occurs, and B, the executions in which $rcv(m)_j$ occurs by time $t + f_{rcv}$. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon_{rcv}$.

- **Acknowledgment delay bound:** Let β be a closed execution that ends with a $bcast(m)_i$ at time t. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m)_i$ occurs, and B, the executions in which $ack(m)_j$ occurs by time $t + f_{ack}$ and is preceded by $rcv(m)_j$ for every

[1] Here and elsewhere, subscripts are used to identify the node at which the event occurs.

[2] An execution of a PTIOA is closed if it is a finite sequence of discrete steps and trajectories, ending with a trajectory whose domain is a right-closed time interval. Formal details of such definitions appear in [12, 21].

neighbor j of i. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon_{ack}$.

- **Progress bound:** Let β be a closed execution that ends at time t. Let I be the set of neighbors of j that have active *bcast*s at the end of β, where $bcast(m_i)_i$ is the bcast at i, and suppose that I is nonempty. Suppose that no $rcv(m_i)_j$ occurs in β, for any $i \in I$. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m_i)_i$ occurs for any $i \in I$, and B, the executions in which, by time $t + f_{prog}$, at least one of the following occurs: an $ack(m_i)_i$ for every $i \in I$, a $rcv(m_i)_j$ for some $i \in I$, or a rcv_j for some packet whose *bcast* occurs after β. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon_{prog}$.

The receive bound says that, with probability at least $1 - \epsilon_{rcv}$, a packet sent by node i is received by a particular neighbor j within time f_{rcv}. The acknowledgment bound says that, with probability at least $1 - \epsilon_{ack}$, a packet sent by node i is acknowledged within time f_{ack}, and moreover, the acknowledgment is "correct" in the sense that the packet has actually been delivered to all neighbors. The progress bound says that, if a nonempty set of j's neighbors have active *bcasts* at some point, and none of these packets has yet been received by j, then with probability at least $1 - \epsilon_{prog}$, within time f_{prog}, either j receives one of these packets or something newer, or else all of these end with acknowledgments. This is all conditioned on non-occurrence of aborts.

6. MAC LAYER ALGORITHM USING ANC

Now we present our new ANC-based MAC-layer algorithm, *CMAC*, and show that it implements the probabilistic MAC layer of Section 5 with certain delay and error parameters. *CMAC* yields smaller receive and acknowledgment delay bounds than conventional probabilistic transmission protocols such as the *DMAC* algorithm in [14, 13]. Its progress bound, on the other hand, is larger. In Section 7, we combine *CMAC* and *DMAC* to obtain a small progress bound as well.

6.1 The CMAC Algorithm

The *CMAC* algorithm is based on the *Coding* algorithm of Section 4.

> **CMAC**(ϵ), where $0 < \epsilon < 1$: We group slots into *Coding* phases, each consisting of $\mathcal{R}_\epsilon$ slots. At the beginning of every *Coding* phase, each node i that has an active $bcast(m)_i$ participates in $Coding(\epsilon)$ with packet m. Node i executes exactly one *Coding* phase, and then outputs $ack(m)_i$ at the end of the phase. However, if node i receives an $abort(m)_i$ from the environment before it performs $ack(m)_i$, it continues participating in the rest of the *Coding* phase but does not perform $ack(m)_i$.
> Meanwhile, node i tries to receive packets from its neighbors, in every slot. It may receive a packet directly, without any collisions, or indirectly, by decoding collisions. When it receives any packet m' from a neighbor for the first time, it delivers that to the environment with a $rcv(m')_i$ event, at a real time before the time marking the end of the slot.

Note that, in a single slot, node i may receive several packets and deliver them to the environment, by decoding a collection of received collisions. Also note that node i may continue processing a packet for some time after it is aborted; thus, *CMAC* handles aborts differently from *DMAC*. Besides increasing the t_{abort} bound, this way of handling aborts introduces the possibility that the environment may submit a new packet while node i is still transmitting on behalf of the aborted one. According to the rules of *CMAC*, node i will begin handling the new packet at the start of the next *Coding* phase.

We now give five lemmas expressing the properties of $CMAC(\epsilon)$. These lemmas are analogous to some in [13]. The "executions" referred to here are executions of the composition $CMAC \| Env \| Net$, for an arbitrary environment *Env*. First, the non-probabilistic properties are satisfied:

LEMMA 5. *In every execution, the* Proximity, No duplicate receives, No receives after acks, No duplicate acks, *and* No acks after aborts *conditions are satisfied. Also, no rcv happens more than time $\mathcal{R}_\epsilon$ after a corresponding abort.*

PROOF. Straightforward. □

The next lemma provides an absolute bound on acknowledgment time.

LEMMA 6. *In every time-unbounded execution α, the following holds. Consider any $bcast(m)_i$ event in α, and suppose that α contains no $abort(m)_i$. Then an $ack(m)_i$ occurs by the end of the next* Coding *phase that begins after the $bcast(m)_i$.*

PROOF. Immediate from the definition of *CMAC* □

The remaining properties are probabilistic. For these lemmas, we fix any environment *Env* and consider probabilities with respect to the unique probability distribution on executions of $CMAC \| Env \| Net$. The first probabilistic lemma, which is analogous to Lemma 5.5 in [13], bounds the receive delay. Its proof uses our result about *Coding*, Lemma 4.

LEMMA 7. *Let $i, j \in V$, i a neighbor of j. Let β be a closed execution that ends with a $bcast(m)_i$ event. Let cp be the first* Coding *phase that starts strictly after the $bcast(m)_i$. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m)_i$ occurs, and B, the executions in which, by the end of coding phase cp, a $rcv(m)_j$ occurs. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon$.*

PROOF. Assume A, that is, no $abort(m)_i$ occurs. Let I be the set of neighbors of j participating in *Coding* phase *cp*. Since no $abort(m)_i$ occurs, $i \in I$, and so $|I| \geq 1$. Then, Lemma 4 implies that, with probability at least $1 - \epsilon$, a rcv_j for every packet $m_{i'}$, $i' \in I$ occurs in phase *cp*. In particular, $rcv(m)_j$ occurs. Therefore, $Pr_\beta(B|A) \geq 1 - \epsilon$, as needed. □

The second probabilistic lemma is analogous to Lemma 5.6 in [13]. It bounds the acknowledgment delay and gives a guarantee that acknowledgments are preceded by receives.

LEMMA 8. *Let $i \in V$. Let β be any closed prefix of a time-unbounded execution that ends with a $bcast(m)_i$ event. Let cp be the first* Coding *phase that starts strictly after the $bcast(m)_i$. Define the following sets of time-unbounded executions that extend β: A, the executions in which no*

$abort(m)_i$ occurs, and B, the executions in which, by the end of coding phase cp, $ack(m)_i$ occurs and is preceded by $rcv(m)_j$ for every neighbor j of i. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon\Delta$.

PROOF. Lemma 6 implies that $ack(m)_i$ occurs by the end of phase *cp*. For the $rcv(m)_j$ events, by Lemma 7, the probability that each individual $rcv(m)_j$ event occurs by the end of *cp* is at least $1 - \epsilon$. Then, using a union bound, the probability that all the $rcv(m)_j$ events occur by the end of *cp* is at least $1 - \epsilon\Delta$. □

The third probabilistic lemma is analogous to Lemma 5.4 in [13]. It gives a probabilistic bound for progress.

LEMMA 9. *Let $j \in V$ and β be a closed execution that ends at time t. Let I be the set of neighbors of j that have active bcasts at the end of β, where $bcast(m_i)_i$ is the bcast at i. Suppose that I is nonempty. Suppose that no $rcv(m_i)_j$ occurs in β, for any $i \in I$. Let cp be the first* Coding *phase that starts strictly after time t.*
Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m_i)_i$ occurs for any $i \in I$; B, the executions in which, by the end of cp, at least one of the following occurs: a $rcv(m_i)_j$ for some $i \in I$, or a rcv_j for some packet whose bcast occurs after β; and C, the executions in which, by the end of cp, $ack(m_i)_i$ occurs for every $i \in I$.
If $Pr_\beta(A) > 0$, then $Pr_\beta(B \cup C|A) \geq 1 - \epsilon$.

PROOF. As shown in the proof of Lemma 5.4 in [13],

$$Pr_\beta(B \cup C|A) \geq Pr_\beta(B|\bar{C} \cap A),$$

so, for the first conclusion, it suffices to show that $Pr_\beta(B|\bar{C} \cap A) \geq 1-\epsilon$. Thus, assume $\bar{C} \cap A$, that is, that by the end of *cp*, not every $i \in I$ has an $ack(m_i)_i$, and no $abort(m_i)_i$ occurs for any $i \in I$. Then some neighbor of j in I participates in phase *cp*. Let I' be the set of neighbors of j participating in *cp*. Note that every node in I' participates in all slots of phase *cp*, since no node stops participating part-way through the phase. Then $|I'| \geq 1$ and thus by Lemma 4, with probability at least $1 - \epsilon$, a rcv_j for every packet $m_{i'}$, $i' \in I'$ occurs in phase $\mathcal{P}$. Therefore,

$$Pr_\beta(B|\bar{C} \cap A) \geq 1 - \epsilon,$$

as needed. □

6.2 Implementing the Probabilistic MAC

Using the lemmas from Section 6.1, we can now show that *CMAC* implements the probabilistic MAC layer with certain parameter values. In this subsection, we fix ϵ, $0 < \epsilon < 1$, and fix t_{Cphase}, the time for a *Coding* phase, to be $\mathcal{R}_\epsilon$, as defined in Section 4.2.

THEOREM 10. *$CMAC(\epsilon)$ implements the probabilistic abstract MAC layer with parameters $f_{rcv} = f_{ack} = f_{prog} = 2t_{Cphase}$, $\epsilon_{rcv} = \epsilon_{prog} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, and $t_{abort} = t_{Cphase}$.*

PROOF. Similar to the proof of Theorem 5.7 in [13], using Lemmas 5-9. □

The following corollary follows directly from Lemma 3 and Theorem 10.

COROLLARY 11. *$CMAC(\epsilon)$ implements the probabilistic MAC layer with time bounds f_{rcv}, f_{ack}, f_{prog}, and t_{abort} equal to*

$$\mathcal{O}\left(\Delta + \left\lceil \frac{\Delta}{c} \right\rceil \log \frac{\Delta}{\epsilon}\right),$$

where $\epsilon_{rcv} = \epsilon_{prog} = \epsilon$ and $\epsilon_{ack} = \epsilon\Delta$.

In some cases, where the threshold c is fairly large and ϵ is not too small, this bound can be simplified as follows.

COROLLARY 12. *Suppose that $c = \Omega(\log n)$, $\Delta = \Omega(\log n)$, and $\epsilon \geq n^{-\kappa}$ for some constant κ. Then $CMAC(\epsilon)$ implements the probabilistic MAC layer with f_{rcv}, f_{ack}, f_{prog}, and $t_{abort} = \mathcal{O}(\Delta)$ $\epsilon_{rcv} = \epsilon_{prog} = \epsilon$, and $\epsilon_{ack} = \epsilon\Delta$.*

Similar bounds hold in the case where c is large compared to Δ:

COROLLARY 13. *If $c = \Omega(\Delta)$ then $CMAC(\epsilon)$ implements the probabilistic MAC layer with f_{rcv}, f_{ack}, f_{prog}, and $t_{abort} = \mathcal{O}(\Delta + \log \frac{1}{\epsilon})$, $\epsilon_{rcv} = \epsilon_{prog} = \epsilon$, and $\epsilon_{ack} = \epsilon\Delta$.*

For comparison, the *DMAC* algorithm [14, 13] yields larger bounds of $f_{rcv} = f_{ack} = \mathcal{O}\big(\Delta \log\left(\frac{1}{\epsilon}\right) \log \Delta\big)$, with $\epsilon_{rcv} = \epsilon$ and $\epsilon_{ack} = \epsilon\Delta$. However, *DMAC* yields a smaller f_{prog} bound, of $\mathcal{O}(h \log \Delta)$, with $\epsilon_{prog} = (\frac{7}{8})^h$, for any positive integer h. In the next section, we show how to reduce the f_{prog} bound to this level, while keeping the other bounds as they are for *CMAC*.

7. IMPROVED MAC LAYER ALGORITHM

Our MAC layer implementation in Section 6 achieves good f_{rcv} and f_{ack} bounds compared to *DMAC* but a worse f_{prog} bound. Now we describe a second MAC layer implementation that achieves both the $\mathcal{O}\left(\Delta + \frac{\Delta}{c} \log \frac{\Delta}{\epsilon}\right)$ receive and acknowledgment bounds of *CMAC*, as well as the $\mathcal{O}(\log \Delta)$ progress bound of *DMAC*. The new algorithm essentially combines *CMAC* and *DMAC* using time-division multiplexing. *CMAC* is used to guarantee the receive and acknowledgment bounds, while *DMAC* guarantees the progress bound. We call the combined algorithm *DCMAC*.

7.1 The DCMAC Algorithm

Technically, *DCMAC* applies the *Coding* subroutine described in Section 4.2 and the *Decay* subroutine of [14, 13]. *Decay* operates for exactly $\sigma = \lceil \log(\Delta + 1) \rceil$ slots, in which participating nodes transmit with successively doubling probabilities, starting with $\frac{1}{2^\sigma}$ and ending with $\frac{1}{2}$.

> **DCMAC**(ϵ), where $0 < \epsilon < 1$: We use odd-numbered slots for *Decay* and even-numbered slots for *Coding*(ϵ). We group odd slots into *Decay* phases, each consisting of σ slots, and group even slots into *Coding* phases, each consisting of $\mathcal{R}_\epsilon$ slots. The two types of phases are not synchronized with respect to each other.
> At the beginning of each *Decay* phase, each node i that has an active $bcast(m)_i$ begins executing *Decay* with packet m. At the beginning of each *Coding* phase, each node i that has an active $bcast(m)_i$ begins executing *Coding*(ϵ) with packet m and outputs $ack(m)_i$ at the end of that *Coding* phase.

Meanwhile, node If node i receives an $abort(m)_i$ or performs an $ack(m)_i$, it performs no further transmission on behalf of packet m in the odd slots; that is, it stops participating in a *Decay* phase as soon as an *abort* or *ack* happens. However, if node i receives an $abort(m)_i$ before it performs $ack(m)_i$, it continues participating in the rest of the *Coding* phase and does not perform $ack(m)_i$.

i keeps trying to receive, in every slot. In even slots, it may receive a packet directly, without collisions, or indirectly, by decoding collisions. In odd slots, it does not try to decode collisions, but just looks for packets that arrive directly. When node i receives any packet m' for the first time, in either an odd or even slot, it delivers that to its environment with a $rcv(m')_i$ event, at a real time before the time marking the end of the slot.

Thus, as in *CMAC*, node i may receive several packets in one slot, by decoding a collection of received collisions. Also note that node i may handle two different packets in consecutive odd and even slots, because of the different handling of aborts in the odd and even slots. However, i handles at most one packet in each slot, odd or even.

As for *CMAC*, we give five lemmas expressing the properties of *DCMAC*, now in terms of executions of the composition $DCMAC \| Env \| Net$. The first four lemmas are similar to their counterparts in Section 6, The fifth lemma, which deals with the progress bound, is somewhat different because it depends on *Decay* rather than *Coding*.

LEMMA 14. *In every execution, the* Proximity, No duplicate receives, No receives after acks, No duplicate acks, *and* No acks after aborts *conditions are satisfied. Also, no rcv happens more than time $2\mathcal{R}_\epsilon$ after a corresponding abort.*

PROOF. Straightforward. □

LEMMA 15. *In every time-unbounded execution α, the following holds. Consider any $bcast(m)_i$ event in α and suppose that α contains no $abort(m)_i$. Then an $ack(m)_i$ occurs at the end of the* Coding *phase that begins after the $bcast(m)_i$.*

PROOF. Immediate from the definition of *DCMAC*. □

The remaining properties are probabilistic. Fix any environment *Env* and consider the unique probability distribution on executions of $DCMAC \| Env \| Net$. The first probabilistic lemma bounds the receive delay, and the second bounds the acknowledgment delay and gives a probabilistic guarantee that acknowledgments are preceded by receives. The proofs are similar to those of Lemmas 7 and 8.

LEMMA 16. *Let $i, j \in V$, i a neighbor of j. Let β be a closed execution that ends with a $bcast(m)_i$ event. Let cp be the first* Coding *phase that starts strictly after the $bcast(m)_i$. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m)_i$ occurs, and B, the executions in which, by the end of coding phase cp, a $rcv(m)_j$ occurs. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon$.*

LEMMA 17. *Let $i \in V$ and β be any closed prefix of a time-unbounded execution that ends with a $bcast(m)_i$ event. Further, let cp be the first* Coding *phase that starts strictly after $bcast(m)_i$. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m)_i$ occurs, and B, the executions in which, by the end of coding phase cp, $ack(m)_i$ occurs and is preceded by $rcv(m)_j$ for every neighbor j of i. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon\Delta$.*

The final lemma gives the progress bound.

LEMMA 18. *Let $j \in V$ and h be a positive integer. Let β be a closed execution that ends at time t. Let I be the set of neighbors of j that have active bcasts at the end of β, where $bcast(m_i)_i$ is the bcast at i. Suppose that I is nonempty. Suppose that no $rcv(m_i)_j$ occurs in β, for any $i \in I$.*
Let dp be the hth Decay *phase that starts strictly after time t. Define the following sets of time-unbounded executions that extend β: A, the executions in which no $abort(m_i)_i$ occurs for any $i \in I$; B, the executions in which, by the end of Decay phase dp, at least one of the following occurs: a $rcv(m_i)_j$ for some $i \in I$, or a rcv_j for some packet whose bcast occurs after β; and C, the executions in which, by the end of Decay phase dp, $ack(m_i)_i$ occurs for every $i \in I$. If $Pr_\beta(A) > 0$, then $Pr_\beta(B \cup C|A) \geq 1 - (7/8)^h$.*

PROOF. Analogous to that of Lemma 5.4 in [13]. □

7.2 Implementing the Probabilistic MAC

Using the lemmas from Section 7.1, we can now show that *DCMAC* implements the probabilistic MAC layer with certain parameter values. Fix ϵ, $0 < \epsilon < 1$, and fix t_{Dphase}, the time for a *Decay* phase, to be $\sigma = \lceil \log(\Delta + 1) \rceil$. Let h be any positive integer.

THEOREM 19. *$DCMAC(\epsilon)$ implements the probabilistic MAC layer with parameters $f_{rcv} = f_{ack} = 4t_{Cphase}$, $f_{prog} = 2(h+1)t_{Dphase}$, $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, $\epsilon_{prog} = (\frac{7}{8})^h$, and $t_{abort} = 2t_{Cphase}$.*

PROOF. Similar to the proof of Theorem 5.7 in [13], using Lemmas 14-18. □

The following corollary follows directly from Lemma 3 and Theorem 19.

COROLLARY 20. *$DCMAC(\epsilon)$ implements the probabilistic MAC layer with f_{rcv}, f_{ack}, and t_{abort} equal to*

$$\mathcal{O}\left(\Delta + \left\lceil \frac{\Delta}{c} \right\rceil \log \frac{\Delta}{\epsilon}\right)$$

and $f_{prog} = \mathcal{O}(h \log \Delta)$, where $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, and $\epsilon_{prog} = (\frac{7}{8})^h$.

Again, we specialize the bound to the case where c and Δ are sufficiently large and ϵ is at most polynomially small in n, as well as for the case where c is large compared to Δ.

COROLLARY 21. *Suppose that $c = \Omega(\log n)$, $c = \Omega(\log n)$, and $\epsilon \geq n^{-\kappa}$ for some constant κ. Then, $f_{rcv} = f_{ack} = \mathcal{O}(\Delta)$, $f_{prog} = \mathcal{O}(h \log(\Delta))$, $\epsilon_{prog} = (\frac{7}{8})^h$, $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, and $t_{abort} = \mathcal{O}(\Delta)$.*

COROLLARY 22. *If $c = \Omega(\Delta)$, $DCMAC(\epsilon)$ implements the probabilistic MAC layer with f_{rcv}, f_{ack}, and $t_{abort} = \mathcal{O}(\Delta + \log \frac{1}{\epsilon})$, $f_{prog} = \mathcal{O}(h \log \Delta)$, $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, and $\epsilon_{prog} = (\frac{7}{8})^h$.*

These bounds compare favorably in all dimensions with those of *DMAC*.

8. NETWORK-WIDE BROADCAST

In addition to defining the probabilistic abstract MAC layer specification and providing the *DMAC* implementation of the specification, the earlier papers by Khabbazian, *et al.* [14, 13] describe and analyze single-message and multi-message network-wide broadcast protocols over the probabilistic MAC layer. The authors show how to combine such high-level protocols with the *DMAC* implementation to obtain efficient protocols for network-wide broadcast over a collision-prone radio network. In fact, a main point of those papers is that one can use abstract MAC layer specifications to split up the task of designing efficient high-level protocols for the radio network model.

Since we use the same probabilistic MAC layer specification as in [14, 13], we are now able to combine the network-wide broadcast protocol from [14, 13] with our new MAC implementations, to obtain efficient network-wide broadcast protocols for physical networks supporting Analog Network Coding. The results follow as easy corollaries of the results so far in this paper and the high-level analysis results in [14, 13].

For instance, consider the problem of Multi-Message Broadcast (MMB). In this problem, an arbitrary number of uniquely-identified messages originate at arbitrary nodes in the network, at arbitrary times; the problem is to deliver all messages to all nodes. For simplicity, we assume that each message fits in a single MAC-layer packet.

In our formulation, an MMB protocol has an external interface by which it receives messages from its environment via $arrive(m)$ input events and delivers messages to the environment via $deliver(m)$ output events. The Basic Multi-Message Broadcast ($BMMB$) protocol from [16, 17] is a greedy protocol, which works as follows:

> **Basic Multi-Message Broadcast Protocol (BMMB):** Every node i maintains a FIFO queue named *bcastq* and a set named *rcvd*. Both are initially empty. If node i does not have a pending MAC-layer transmission and *bcastq* is not empty, node i composes a packet containing the message m at the head of *bcastq*, removes m from *bcastq*, and passes the packet to the MAC layer for transmission, using a *bcast* output event. If node i receives an $arrive(m)$ input event, it immediately performs a $deliver(m)$ output and adds m to the back of *bcastq* and to the *rcvd* set. If node i receives a message m from the MAC layer, it first checks *rcvd*. If $m \in rcvd$ it discards the message. Otherwise, node i immediately performs a $deliver(m)$ output, and adds m to the back of *bcastq* and to the *rcvd* set.

We now consider executions of *BMMB* composed with *DCMAC* and an environment that generates the messages. Theorem 23 provides a probabilistic bound on the time for a message to be delivered to all nodes, in the presence of a bounded number k' of concurrent messages. This theorem uses two definitions from [13]:

Definition 2 (Nice executions). *A $bcast(m)$ event that occurs at node i at time t_0 in an execution is* nice *if the corresponding $ack(m)$ event occurs by time $t_0 + f_{ack}$ and is preceded by a corresponding $rcv(m)$ event at every neighbor j of i. An execution is* nice *if all bcast events in the execution are nice.*

Definition 3 (The set $overlap(m)$). *Let α be a nice execution and m be a message such that $arrive(m)$ occurs in α. Then we define $overlap(m)$ to be the set of messages m' whose processing overlaps the interval between the $arrive(m)$ and the final $ack(m)$ event for m anywhere in the network. Formally, this means that an $arrive(m')$ event precedes the final $ack(m)$ event and the final $ack(m')$ event follows the $arrive(m)$ event.*

The following theorem follows from Theorem 8.20 of [13] and Corollary 20. It assumes an upper bound k on the total number of messages that arrive from the environment in any execution.

Theorem 23. *Let m be a message and ϵ be a real, $0 < \epsilon < 1$. Then $BMMB$ composed with $DCMAC(\frac{\epsilon}{2nk\Delta})$ guarantees that, with probability at least $1 - \epsilon$, the following property holds of the generated execution α: Suppose an $arrive(m)_i$ event occurs in α. Let $k' = |overlap(m)|$. Then $deliver(m)$ events occur at all nodes in α within time*

$$\mathcal{O}\left(\left(D + \log\left(\frac{nk}{\epsilon}\right)k'\right)\log\Delta + (k'-1)\left(\Delta + \frac{\Delta}{c}\log\frac{\Delta nk}{\epsilon}\right)\right).$$

For comparison, the corresponding result for $BMMB$ composed with *DMAC* ([13], Theorem 8.21, paraphrased slightly) is:

Theorem 24. *Let m be a message and ϵ be a real, $0 < \epsilon < 1$. Then $BMMB$ composed with $DMAC$ guarantees that, with probability at least $1 - \epsilon$, the following property holds of the generated execution α: Suppose an $arrive(m)_i$ event occurs in α. Let $k' = |overlap(m)|$. Then $deliver(m)$ events occur at all nodes in α by time*

$$\mathcal{O}\left(\left(D + \Delta\log\left(\frac{nk}{\epsilon}\right)k'\right)\log\Delta\right).$$

Proof (of Theorem 23). The proof is similar to the one for Theorem 24 in [13]. Choose $\epsilon_{ack} = \frac{\epsilon}{2nk}$, so $1 - \frac{\epsilon}{2} - nk\epsilon_{ack} = 1 - \epsilon$. Then we obtain a time bound of $\mathcal{O}((D + log(\frac{nk}{\epsilon})k')f_{prog}) + (k'-1)f_{ack}$.

Next, we plug in bounds for f_{prog} and f_{ack}, based on the bounds for *DCMAC* in Corollary 20. We have $f_{prog} = \mathcal{O}(\log\Delta)$, which yields the first term of the claimed bound. For f_{ack}, we instantiate ϵ in Corollary 20 with $\frac{\epsilon_{ack}}{\Delta} = \frac{\epsilon}{2nk\Delta}$, and obtain $f_{ack} = \mathcal{O}(\Delta + \frac{\Delta}{c}log(\frac{2nk\Delta^2}{\epsilon}))$, which is $\mathcal{O}(\Delta + \frac{\Delta}{c}log(\frac{\Delta nk}{\epsilon}))$. This yields the second term of the claimed bound. □

9. CONCLUSIONS

We have presented a MAC layer algorithm, *CMAC*, based on Analog Network Coding. We have proved its basic correctness and performance properties by showing that *CMAC* implements a formal probabilistic abstract MAC layer specification. This analysis shows that the new design improves on a conventional probabilistic retransmission algorithm, like *DMAC*, in two of three performance metrics (the receive and acknowledgment delay bounds), while doing worse on one (the progress bound). However, a hybrid design, *DCMAC*, which combines *CMAC* and *DMAC*, achieves the best of both algorithms.

In addition to providing an objective basis for comparing MAC layer designs, the abstract MAC layer allows us

to combine complexity bounds for MAC layer designs with complexity bounds for higher-level protocols that run over MAC layers. To illustrate this, we showed how to combine a network-wide broadcast protocol, *BMMB*, with an ANC-based MAC layer, and easily obtain complexity bounds for the combination.

There are many possible directions for future work. First, we would like to understand whether the particular transmission strategies used in *CMAC* and *DCMAC* are optimal, and if not, how they can be improved. We would like to extend the algorithms and results to accommodate packet losses. We would also like to compare ANC-based MAC-layer strategies with more different kinds of MAC-layer designs. Also, our MAC layer specifications provide three kinds of latency bounds; we would also like to extend the work to consider other metrics, such as throughput.

Network coding provides a much richer set of strategies than what we have used in this paper; we would like to extend our theory to take advantage of more of these strategies. For example, our development of the MAC has considered physical-layer ANC and not higher-layer network coding. The use of our MAC and related approaches could be considered in the presence of transport-layer network coding, since the two network coding approaches are compatible [22]. Also, our work here has considered coding only for local node interactions, but it invites questions of considering it for larger, multihop networks, particularly when transport-layer coding is integrated. The interaction between the broadcast MAC and network coding at the transport layer has been shown to provide, in a simple, opportunistic fashion, considerable gains in a multihop setting [11]. Moreover, a MAC-aware coding approach in multihop networks can lead to even more considerable gains [25]. It remains to extend our theory to incorporate these new factors.

10. REFERENCES

[1] M. N. A. Tehrani, A. Dimakis. SigSag: Iterative detection through soft message-passing. In *Proc. of IEEE INFOCOM*, 2011.

[2] N. Abramson. ALOHA System-Another alternative for computer communications. In *Proc. of Fall Joint Computer Conf.*, 1970.

[3] D. Bertsekas and R. Gallager. *Data networks (2nd ed.)*. Upper Saddle River, NJ, US: Prentice-Hall, Inc., 1992.

[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A medium access protocol for wireless lans. In *Proc. of ACM SIGCOMM*, 1994.

[5] H. Bolcskei, R. U. Nabar, O. Oyman, , and A. J. Paulraj. Capacity scaling laws in MIMO relay networks. *IEEE Transactions on Wireless Communications*, 5(6):1433–1443, 2006.

[6] S. Borade, L. Zheng, and R. Gallager. Amplify-and-forward in wireless relay networks: Rate, diversity and network size. *IEEE Transactions on Information Theory*, 53(10):3302–3318, 2007.

[7] A. Cornejo, N. Lynch, S. Viqar, and J. Welch. A neighbor discovery service using an abstract MAC layer. In *Proc. of the Allerton Conf. on Communication, Control, and Computing*, pages 1460–1467, 2009.

[8] A. Cornejo, S. Viqar, and J. Welch. Reliable neighbor discovery for mobile ad hoc networks. In *Proc. of 6th DIALM-POMC*, pages 63–72, 2010.

[9] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *Proc. of ACM SIGCOMM*, pages 159–170, 2008.

[10] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *Proc. of ACM SIGCOMM*, pages 397–408, 2007.

[11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, 2008.

[12] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *Theory of Timed I/O Automata, Second Edition*. Synthesis Lectures on Computer Science. Morgan-Claypool Publishers, 2010.

[13] M. Khabbazian, D. Kowalski, F. Kuhn, and N. Lynch. Decomposing global broadcast algorithms using abstract MAC layers. *MIT Technical Report (MIT-CSAIL-TR-2011-010)*, February 2011.

[14] M. Khabbazian, F. Kuhn, D. Kowalski, and N. Lynch. Decomposing broadcast algorithms using abstract MAC layers. In *Proc. of 6th DIALM-POMC*, pages 13–22, 2010.

[15] M. Khabbazian, F. Kuhn, N. Lynch, M. Medard, and A. Parandeh-Gheibi. Mac design for analog network coding. *MIT Technical Report (MIT-CSAIL-TR-2010-036)*, July 2010.

[16] F. Kuhn, N. Lynch, and C. Newport. The abstract MAC layer. *MIT Technical Report (MIT-CSAIL-TR-2009-021)*, May 2009.

[17] F. Kuhn, N. Lynch, and C. Newport. The abstract MAC layer. In *Proc. of the Int. Symp. on Distributed Computing*, pages 48–62, 2009.

[18] F. Kuhn, N. Lynch, and C. Newport. The abstract MAC layer. *To appear in special issue of Distributed Computing*, 2010.

[19] I. Maric, A. Goldsmith, and M. Médard. Analog network coding in the high SNR regime. *invited paper, ITA Workshop*, 2010.

[20] I. Maric, A. Goldsmith, and M. Médard. Analog network coding in the high SNR regime. In *Proc. of Wireless Network Coding Workshop*, 2010.

[21] S. Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[22] A. ParandehGheibi, J. Sundararajan, and M. Médard. Collision helps - algebraic collision recovery for wireless erasure networks. In *Proc. of Wireless Network Coding Workshop*, 2010.

[23] D. Wischik and D. Shah. MAC3: Medium access coding & congestion control. *Presentation at the Newton Institute for Mathematics*, 2010.

[24] S. Zhang, S. Liew, , and P. Lam. Hot topic: Physical layer network coding. In *Proc. of ACM MOBICOM*, pages 358–365, 2006.

[25] F. Zhao and M. Médard. On analyzing and improving COPE performance. In *ITA Workshop*, January 2010.