
As Published: http://dx.doi.org/10.1145/1998476.1998484

Publisher: Association for Computing Machinery (ACM)

Persistent URL: http://hdl.handle.net/1721.1/72944

Version: Author’s final manuscript: final author’s manuscript post peer review, without publisher’s formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike 3.0
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

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 \( O(\Delta + \frac{1}{\epsilon} \log \Delta) \), where \( \Delta \) is the maximum node degree and \( \epsilon \) is the maximum number of packets for which a collision can be decoded. In particular, if \( \epsilon \) 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 \( O(\Delta + \log \frac{1}{\epsilon}) \).

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 \( O(\Delta \log (\frac{1}{\epsilon}) \log \Delta) \), which are larger than those of CMAC, but have a smaller progress bound, of the form \( O(\log \Delta) \). In Section 7, we present another MAC protocol, DCMAC, which improves the progress bound of CMAC to \( O(\log \Delta) \) 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 protocols 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 \( C \) 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, \( \Delta \) 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 \( \Delta \) 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 $l$. 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 basis). We abstract the modulation to be a one-to-one map $M$ from symbols over $\mathbb{F}_q$ to the complex number field

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

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

$$\begin{pmatrix} C^s_{ij}(k) \\ C^w_{ij}(k) \end{pmatrix} = \begin{pmatrix} A^1_{ij} & \cdots & A^w_{ij} \\ \vdots & \ddots & \vdots \\ A^1_{ij} & \cdots & A^w_{ij} \end{pmatrix} \begin{pmatrix} S^1_{ij}(k) \\ \vdots \\ S^w_{ij}(k) \end{pmatrix},$$

$k = 1, \ldots, N$, where $C^s_{ij} \in \mathbb{C}^N$ represents the collision in time slot $s$, $S^i \in \mathbb{C}^N$ is the signal (packet) transmitted by sender $i$, and $A^s_{ij} \in \mathbb{C}$ 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 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

$$A = \begin{pmatrix} A^1_{11} & A^1_{12} & \cdots & A^1_{1w} \\ A^2_{11} & A^2_{12} & \cdots & A^2_{1w} \\ \vdots & \vdots & \ddots & \vdots \\ A^w_{11} & A^w_{12} & \cdots & A^w_{1w} \end{pmatrix}.$$

**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 $k = 1, 2, \ldots, N$ 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 et 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 $4 \leq c \leq \Delta$. Note that, based on our network assumptions, at most $\Delta$ 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 $B$ 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 $\ell$ 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 $\epsilon$ be real numbers, with $0 < p \leq \frac{1}{2}$ and $0 < \epsilon < 1$. Let $c$ be a positive integer with $c \geq 4$ and $wp \leq \frac{1}{2}$. Suppose that

$$\ell \geq \left\lceil \frac{4w}{1-p} \left( w + \frac{\ln(w+1) + 2\ln(1/\epsilon)}{p} \right) \right\rceil.$$  

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 $B$, we assume that we construct $B$ by adding rows until there are $w$ rows with at most $c$ non-zero entries that span $\mathbb{C}^w$. The lemma then follows by showing that with probability at least $1 - \epsilon$, the resulting matrix has at most $\ell$ 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 $x_1, \ldots, x_{d-1}$ for some integer $d \geq 1$. 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 $x_i$ 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 $x_i$ 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 new non-zero 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 $x_1, \ldots, x_{d-1}$ is at least

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

Further it can be shown that the same bound holds if the vectors $x_1, \ldots, x_{d-1}$ are arbitrary vectors in $\mathbb{C}^w$. Therefore, for each $d \geq 0$ 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

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

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 $\rho = \frac{\pi}{2\kappa}$. Note that $\rho \leq \frac{1}{7}$ because $\epsilon \leq \Delta$.

**Definition 1.** ($\mathcal{R}_\epsilon$, where $\epsilon$ is a real, $0 < \epsilon < 1$).

$$\mathcal{R}_\epsilon = \left[ \frac{1}{1 - \rho} \left( \Delta + \frac{\ln(\Delta + 1) + 2\ln(1/\epsilon)}{\rho} \right) \right].$$

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

The Coding algorithm has a single explicit parameter, $\epsilon$. It also uses $c$ as an implicit parameter.

**Coding($\epsilon$), where $\epsilon$ is a real, $0 < \epsilon < 1$: Assume $I$ is a set of nodes, $1 \leq |I| \leq \Delta$, 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 $\rho = \frac{\pi}{2\kappa}$.

**Lemma 4.** In Coding($\epsilon$), 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 $1$, $1 \leq r \leq \mathcal{R}_\epsilon$, 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 $I = \{i_s | 1 \leq s \leq |I| \}$. For every $s$, $1 \leq s \leq |I|$, let $B_{r,s} = 0$ if node $i_s$ does not transmit.
Since acknowledgment to be returned to a sender, and for some positive reals, input events.

Coding and finally, it has a parameter \( t \) probabilities that the delay bounds are not attained. For

parameters \( \epsilon \) receiver. The specification also has corresponding parameters \( \rho \) vs. every \( m \), in the specification of \( \text{Coding}(\epsilon) \). 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 \( \text{MACM} \), 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 \( \text{bcast}(m) \) input events and delivers packets to neighboring nodes via \( \text{rcv}(m) \) output events. It also provides acknowledgments to senders indicating that their packets have been successfully delivered to all neighbors, via \( \text{ack}(m) \) output events. Finally, it accepts requests from the environment to abort current broadcasts, via \( \text{abort}(m) \) input events.

The specification is implicitly parameterized by three positive reals, \( f_{\text{rcv}}, f_{\text{ack}}, \) and \( f_{\text{prop}} \). 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 \( \epsilon_{\text{prop}}, \epsilon_{\text{rcv}}, \) and \( \epsilon_{\text{ack}} \), which represent bounds on the probabilities that the delay bounds are not attained. Finally, it has a parameter \( t_{\text{abort}} \), 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 \( \text{Mac||Env||Net} \), 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 \( \alpha \) of Mac||Env||Net that are well-formed, in the sense that:

(a) they contain at most one \( \text{bcast}(m) \) event for each \( m \), i.e., all packets are unique,
(b) any \( \text{abort}(m) \),\(^1\) event is preceded by a \( \text{bcast}(m) \), but not by an \( \text{ack}(m) \), or another \( \text{abort}(m) \),
(c) any two \( \text{bcast} \) events have an intervening \( \text{ack} \), 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 \( \alpha \) of Mac||Env||Net. There exists a cause function that maps every \( \text{rcv}(m) \), event \( \alpha \) to a preceding \( \text{bcast}(m) \) event, \( i \neq j \), and that maps each \( \text{ack}(m) \), and \( \text{abort}(m) \), to a preceding \( \text{bcast}(m) \). The cause function must satisfy:

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

- **Acknowledgment restrictions**: If \( \text{bcast}(m_i) \), event \( \alpha \) causes \( \text{ack}(m_j) \), event \( \alpha \), then (a) No duplicate \( \text{ack} \): No other \( \text{ack}(m) \), caused by \( \alpha \) precedes \( \alpha' \). (b) No \( \text{ack} \) after \( \text{abort} \): No \( \text{abort}(m) \), caused by \( \alpha \) precedes \( \alpha' \).

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 \( \alpha \) is a \( \text{bcast}(m) \) event in a closed execution \( \beta \), then we say that \( \alpha \) is active at the end of \( \beta \) provided that \( \alpha \) is not terminated with an \( \text{ack} \) or \( \text{abort} \) in \( \beta \). The probabilistic MAC layer guarantees the following probabilistic bounds. Here, the notation \( P_{\beta}(A) \) refers to the conditional distribution on executions that extend \( \beta \). Assume \( i, j \in V \), and \( t \) is a nonnegative real.

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

- **Acknowledgment delay bound**: Let \( \beta \) be a closed execution that ends with a \( \text{bcast}(m_i) \), at time \( t \). Define the following sets of time-unbounded executions that extend \( \beta \): \( A \), the executions in which no \( \text{abort}(m_i) \), occurs, and \( B \), the executions in which \( \text{ack}(m_j) \), occurs by time \( t + f_{\text{ack}} \) and is preceded by \( \text{rcv}(m_j) \), 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 \(\beta\) be a closed execution that ends at time \(t\). Let \(I\) be the set of neighbors of \(j\) that have active \(\text{beasts}\) at the end of \(\beta\), where \(\text{beast}(m_i)\) is the bcast at \(i\), and suppose that \(I\) is nonempty. Suppose that no \(\text{rcv}(m_i)\) occurs in \(\beta\), for any \(i \in I\). Define the following sets of time-unbounded executions that extend \(\beta\): \(A\), the executions in which no \(\text{abort}(m_i)\) occurs for any \(i \in I\), and \(B\), the executions in which, by time \(t + f_{\text{prog}}\), at least one of the following occurs: an \(\text{ack}(m_i)\), for every \(i \in I\), a \(\text{rcv}(m_i)\) for some \(i \in I\), or a \(\text{rcv}_j\) for some packet whose \(\text{beast}\) occurs after \(\beta\).

If \(Pr_\beta(A) > 0\), then \(Pr_\beta(B|A) \geq 1 - \epsilon_{\text{prog}}\).

The receive bound says that, with probability at least \(1 - \epsilon_{\text{rcv}}\), a packet sent by node \(i\) is received by a particular neighbor \(j\) within time \(f_{\text{rcv}}\). The acknowledgment bound says that, with probability at least \(1 - \epsilon_{\text{ack}}\), a packet sent by node \(i\) is acknowledged within time \(f_{\text{ack}}\), 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 \(\text{beasts}\) at some point, and none of these packets has yet been received by \(j\), then with probability at least \(1 - \epsilon_{\text{prog}}\), within time \(f_{\text{prog}}\), 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, \(\text{CMAC}\), and show that it implements the probabilistic MAC layer of Section 5 with certain delay and error parameters. \(\text{CMAC}\) yields smaller receive and acknowledgment delay bounds than conventional probabilistic transmission protocols such as the \(\text{DMAC}\) algorithm in [14, 13]. Its progress bound, on the other hand, is larger. In Section 7, we combine \(\text{CMAC}\) and \(\text{DMAC}\) to obtain a small progress bound as well.

6.1 The CMAC Algorithm

The \(\text{CMAC}\) algorithm is based on the \(\text{Coding}\) algorithm of Section 4.

\(\text{CMAC}(\epsilon)\), where \(0 < \epsilon < 1\): We group slots into \(\text{Coding}\) phases, each consisting of \(R\) slots. At the beginning of every \(\text{Coding}\) phase, each node \(i\) that has an active \(\text{beast}(m_i)\) participates in \(\text{Coding}(\epsilon)\) with packet \(m\). Node \(i\) executes exactly one \(\text{Coding}\) phase, and then outputs \(\text{ack}(m_i)\) at the end of the phase. However, if node \(i\) receives an \(\text{abort}(m_i)\) from the environment before it performs \(\text{ack}(m_i)\), it continues participating in the rest of the \(\text{Coding}\) phase but does not perform \(\text{ack}(m_i)\).

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 \(\text{rcv}(m'_i)\) 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, \(\text{CMAC}\) handles aborts differently from \(\text{DMAC}\). Besides increasing the \(t_{\text{abort}}\) 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 \(\text{CMAC}\), node \(i\) will begin handling the new packet at the start of the next \(\text{Coding}\) phase.

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

**Lemma 5.** In every execution, the Proximity. No duplicate receives. No receives after acks. No duplicate acks. No acks after aborts conditions are satisfied. Also, no \(\text{rcv}\) happens more than time \(R\), after a corresponding abort.

**Proof.** Straightforward.

The next lemma provides an absolute bound on acknowledgment time.

**Lemma 6.** In every time-unbounded execution \(\alpha\), the following holds. Consider any \(\text{beast}(m_i)\) event in \(\alpha\), and suppose that \(\alpha\) contains no \(\text{abort}(m_i)\). Then an \(\text{ack}(m_i)\) occurs by the end of the next \(\text{Coding}\) phase that begins after the \(\text{beast}(m_i)\).

**Proof.** Immediate from the definition of \(\text{CMAC}\)

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

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

**Proof.** Assume \(A\) that, is no \(\text{abort}(m_i)\), occurs. Let \(I\) be the set of neighbors of \(j\) participating in \(\text{Coding}\) phase \(cp\). Since no \(\text{abort}(m_i)\) occurs, \(i \in I\), and so \(|I| \geq 1\). Then, Lemma 4 implies that, with probability at least \(1 - \epsilon\), a \(\text{rcv}\) for every packet \(m_j, i' \in I\) occurs in phase \(cp\). In particular, \(\text{rcv}(m_j)\) 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 \(\beta\) be any closed prefix of a time-unbounded execution that ends with a \(\text{beast}(m_i)\) event. Let \(\alpha\) be the first \(\text{Coding}\) phase that starts strictly after the \(\text{beast}(m_i)\). Define the following sets of time-unbounded executions that extend \(\beta\): \(A\), 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 \( Pr_{\beta}(B | A) \geq 1 - \epsilon \).

**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 \( 1 - \epsilon \).

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 \( \beta \) be a closed execution that ends at time \( t \). Let \( I \) be the set of neighbors of \( j \) that have active beasts at the end of \( \beta \), where \( beast(m_i) \) is the beast at \( i \). Suppose that \( I \) is nonempty. Suppose that no \( rcv(m_i) \) occurs in \( \beta \), 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 \( \beta \): \( A \), the executions in which no abort\( (m_i) \), occurs for any \( i \in I \); \( B \), the executions in which, by the end of \( cp \), at least one of the following occurs: a \( rcv(m_i) \) for some \( i \in I \), or a \( rcv \) for some packet whose beast occurs after \( \beta \); and \( C \), the executions in which, by the end of \( cp \), \( ack(m_i) \), 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],

\[
Pr_{\beta}(B) \cup C(A) \geq Pr_{\beta}(B) \cap C(A),
\]

so, for the first conclusion, it suffices to show that \( Pr_{\beta}(B) \cap C(A) \geq 1 - \epsilon \). Thus, assume \( C \cap A \), that is, that by the end of \( cp \), not every \( i \in I \) has \( ack(m_i) \), and no abort\( (m_i) \), 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 \( rcv \) for every packet \( m_i \), \( i \in I' \) occurs in phase \( P \).

Therefore,

\[
Pr_{\beta}(B \cap C(A)) \geq 1 - \epsilon,
\]

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 \( \epsilon \), where \( 0 < \epsilon < 1 \), and fix \( t_{\text{phase}} \), the time for a Coding phase, to be \( R_{\epsilon} \), as defined in Section 4.2.

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

**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 \( t_{\text{abort}} \) equal to

\[
O \left( \Delta + \frac{\Delta}{\epsilon} \log \frac{\Delta}{\epsilon} \right),
\]

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

In some cases, where the threshold \( \epsilon \) is fairly large and \( \epsilon \) 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 \( \kappa \). Then CMAC\( (\epsilon) \) implements the probabilistic MAC layer with \( f_{rcv}, f_{ack}, f_{prog}, \) and \( t_{\text{abort}} = O(\Delta) \) \( \epsilon_{rcv} = \epsilon_{prog} = \epsilon \), and \( \epsilon_{ack} = \epsilon \).

Similar bounds hold in the case where \( \epsilon \) is large compared to \( \Delta \).

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

For comparison, the DMAC algorithm [14, 13] yields larger bounds of \( f_{rcv} = f_{ack} = O(\Delta \log \frac{\Delta}{\epsilon^2}) \log \Delta \), with \( \epsilon_{rcv} = \epsilon \) and \( \epsilon_{ack} = \epsilon \). However, DMAC yields a smaller \( f_{prog} \) bound, of \( O(h \log \Delta) \), with \( \epsilon_{prog} = \frac{\epsilon}{\epsilon^2} \), 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 \( O(\Delta + \frac{\Delta}{\epsilon} \log \frac{\Delta}{\epsilon}) \) receive and acknowledgment bounds of CMAC, as well as the \( O(\Delta \log \Delta) \) 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 \( \sigma = \log(\Delta + 1) \) slots, in which participating nodes transmit with successively doubling probabilities, starting with \( \frac{1}{2^\sigma} \) and ending with \( \frac{1}{2} \).

DCMAC\( (\epsilon) \), where \( 0 < \epsilon < 1 \): We use odd-numbered slots for Decay and even-numbered slots for Coding\( (\epsilon) \). We group odd slots into Decay phases, each consisting of \( \sigma \) slots, and group even slots into Coding phases, each consisting of \( R_{\epsilon} \) 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 \( beast(m_i) \), begins executing Decay with packet \( m \). At the beginning of each Coding phase, each node \( i \) that has an active \( beast(m_i) \), begins executing Coding\( (\epsilon) \) with packet \( m \) and outputs \( ack(m) \), at the end of that Coding phase.
Meanwhile, node If node 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 recv($m'$) 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 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|\text{Env}||\text{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 recv happens more than time $2R$, after a corresponding abort.

**Proof.** Straightforward.

Lemma 15. In every time-unbounded execution $\alpha$, the following holds. Consider any beast($m_i$), event in $\alpha$ and suppose that $\alpha$ contains no abort($m_i$). Then an ack($m_i$), occurs on the end of the Coding phase that begins after the beast($m_i$).

**Proof.** Immediate from the definition of DCMAC.

The remaining properties are probabilistic. Fix any environment $\text{Env}$ and consider the unique probability distribution on executions of DCMAC|\text{Env}||\text{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 $\beta$ be a closed execution that ends with a beast($m_i$), event. Let $cp$ be the first Coding phase that starts strictly after the beast($m_i$). Define the following sets of time-unbounded executions that extend $\beta$: A, the executions in which no abort($m_i$), occurs, and $B$, the executions in which, by the end of coding phase $cp$, an ack($m_i$) occurs. If $Pr_\beta(A) > 0$, then $Pr_\beta(B|A) \geq 1 - \epsilon$.

Lemma 17. Let $i \in V$ and $\beta$ be any closed prefix of a time-unbounded execution that ends with a beast($m_i$), event. Further, let $cp$ be the first Coding phase that starts strictly after beast($m_i$). Define the following sets of time-unbounded executions that extend $\beta$: A, the executions in which no abort($m_i$), occurs, and $B$, the executions in which, by the end of coding phase $cp$, an ack($m_i$) occurs and is preceded by recv($m_i$), 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 $\beta$ be a closed execution that ends at time $t$. Let $I$ be the set of neighbors of $j$ that have active beasts at the end of $\beta$, where beast($m_i$), is the beast at $i$.

The remaining properties of aborts in the odd and even slots. However, aborts in the even slots, because of the different handling of aborts in the odd and even slots. No acks after aborts

The proofs are similar to those of Lemmas 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 $\epsilon, 0 < \epsilon < 1$, and fix $t_{\text{phase}}$, the time for a Decay phase, to be $\sigma = [\log(\Delta + 1)]$. Let $h$ be any positive integer.

Theorem 19. DCMAC($\epsilon$) implements the probabilistic MAC layer with parameters $f_{\text{recv}} = f_{\text{ack}} = 4t_{\text{phase}}$, $f_{\text{prog}} = 2(h + 1)t_{\text{phase}}$, $\epsilon_{\text{recv}} = \epsilon$, $\epsilon_{\text{ack}} = \epsilon\Delta$, $\epsilon_{\text{prog}} = \left(\frac{\epsilon}{h}\right)^k$, and $\epsilon_{\text{abort}} = 2t_{\text{phase}}$.

**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 $f_{\text{recv}}$, $f_{\text{ack}}$, and $f_{\text{abort}}$ equal to

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

and $f_{\text{prog}} = O(h \log \Delta)$, where $\epsilon_{\text{recv}} = \epsilon$, $\epsilon_{\text{ack}} = \epsilon\Delta$, and $\epsilon_{\text{prog}} = \left(\frac{\epsilon}{h}\right)^k$.

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

Corollary 21. Suppose that $c = \Omega(\log n)$, $\Delta = \Omega(\log n)$, and $\epsilon \geq n^{-1/2}$ for some constant $\kappa$. Then, $f_{\text{recv}} = f_{\text{ack}} = O(\Delta)$, $f_{\text{prog}} = O(h \log \Delta)$, $\epsilon_{\text{prog}} = \left(\frac{\epsilon}{h}\right)^k$, $\epsilon_{\text{recv}} = \epsilon$, $\epsilon_{\text{ack}} = \epsilon\Delta$, and $\epsilon_{\text{abort}} = O(\Delta)$.

Corollary 22. If $c = \Omega(\Delta)$, DCMAC($\epsilon$) implements the probabilistic MAC layer with $f_{\text{recv}}$, $f_{\text{ack}}$, and $f_{\text{abort}} = O(\Delta + \log \frac{1}{\epsilon\Delta})$, $f_{\text{prog}} = O(h \log \Delta)$, $\epsilon_{\text{recv}} = \epsilon$, $\epsilon_{\text{ack}} = \epsilon\Delta$, and $\epsilon_{\text{prog}} = \left(\frac{\epsilon}{h}\right)^k$.

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 Khаббазиан, 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 event. If node i receives an arrive(m) input event, it immediately performs a deliver(m) output event. If node i receives m from the MAC layer, it first checks rcvd. If m ∈ rcvd it discards the message. Otherwise, node i immediately performs a deliver(m) output event, 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 t₀ in an execution is nice if the corresponding ack(m) event occurs by time t₀ + fack 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 < ϵ < 1. Then BMMB composed with DCMAC guarantees that, with probability at least 1 − ϵ, the following property holds of the generated execution α: Suppose an arrive(m), event occurs in α. Let k' = |overlap(m)|. Then deliver(m) events occur at all nodes in α within time

\[ O\left(\left(D + \log\left(\frac{nk}{\epsilon}\right)k'\right)\log\Delta + (k'-1)\left(\Delta + \frac{c}{2}\log\frac{nk}{\epsilon}\right)\right) \]

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 < ϵ < 1. Then BMMB composed with DMAC guarantees that, with probability at least 1 − ϵ, the following property holds of the generated execution α: Suppose an arrive(m), event occurs in α. Let k' = |overlap(m)|. Then deliver(m) events occur at all nodes in α by time

\[ O\left(\left(D + \Delta \log\left(\frac{nk}{\epsilon}\right)k'\right)\log\Delta\right) \]

**Proof (of Theorem 23).** The proof is similar to the one for Theorem 24 in [13]. Choose ϵ' = \frac{\epsilon}{\Delta + c} \log\left(\frac{nk}{\epsilon}\right)

Next, we plug in bounds for fprog and fack, based on the bounds for DCMAC in Corollary 20. We have fprog = O(\log\Delta), which yields the first term of the claimed bound. For fack, we instantiate ϵ in Corollary 20 with \frac{\Delta + c}{\Delta + c + \Delta} and obtain fack = O(\Delta + \frac{c}{2}\log\left(\frac{nk}{\epsilon}\right)), which is O(\Delta + \frac{c}{2}\log\left(\frac{nk}{\epsilon}\right)). This yields the second term of the claimed bound.

\[ O\left(\left(D + \log\left(\frac{nk}{\epsilon}\right)k'\right)\log\Delta\right) \]

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