Decomposing Broadcast Algorithms Using Abstract MAC Layers

Majid Khabbazian*†
Department of Applied Computer Science
University of Winnipeg, Canada
m.khabbazian@uwinnipeg.ca

Dariusz R. Kowalski‡
Department of Computer Science
University of Liverpool, United Kingdom
d.kowalski@liverpool.ac.uk

Fabian Kuhn*
Faculty of Informatics
University of Lugano (USI), Switzerland
fabian.kuhn@usi.ch

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

ABSTRACT

In much of the theoretical literature on wireless algorithms, issues of message dissemination are considered together with issues of contention management. This combination leads to complicated algorithms and analysis, and makes it difficult to extend the work to harder communication problems. In this paper, we present results of a current project aimed at simplifying such algorithms and analysis by decomposing the treatment into two levels, using abstract "MAC layer" specifications to encapsulate the contention management. We use two different abstract MAC layers: the basic one of [14, 15] and a new probabilistic layer.

We first present a typical randomized contention-manageent algorithm for a standard graph-based radio network model We show that it implements both abstract MAC layers. We combine this algorithm with greedy algorithms for single-message and multi-message global broadcast and analyze the combination, using both abstract MAC layers as intermediate layers. Using the basic MAC layer, we prove a bound of $O(D \log(\frac{n}{\epsilon}) \log \Delta)$ for the time to deliver a single message everywhere with probability $1 - \epsilon$, where D is the network diameter, n is the number of nodes, and Δ is the maximum node degree. Using the probabilistic layer, we prove a bound of $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$, which matches the best previously-known bound for single-message broadcast over the physical network model. For multi-message broadcast, we obtain bounds of $O((D + k\Delta) \log(\frac{n}{\epsilon}) \log \Delta)$ using the basic layer and $O((D + k\Delta \log(\frac{n}{\epsilon})) \log \Delta)$ using the probabilistic layer, for the time to deliver a message everywhere in the presence of at most k concurrent messages.

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*

General Terms

Algorithms, Theory

Keywords

multi-message broadcast, MAC layer, contention management, wireless network algorithms

1. INTRODUCTION

The last few years have seen a rapid growth in analytical work on algorithms for wireless ad hoc networks. This work has generally followed one of two approaches. The first, represented by [19], analyzes algorithms using standard message-passing models, and ignores interference and other low-level communication issues, assuming that they are handled by a separate Medium Access Control (MAC) layer. The second approach, represented by [2], uses models that are close to the actual physical network and requires algorithms to handle basic communication issues as well as high-level issues. Ignoring MAC layer issues and working with high-level communication models makes it possible to design and analyze interesting algorithms for high-level problems. However, such analysis may not be realistic: in wireless networks, all nodes share the same wireless medium, which means that, in reality, only a limited amount of information can be transmitted per time unit in a local region. Consequently, analyzing algorithms using classical message-passing models often yields time bounds that are far too optimistic. Designing algorithms directly for the physical network avoids these problems, but requires the algorithm designer to cope with physical layer issues such as message loss due to interference and

*Research supported by AFOSR contract FA9550-08-1-0159 and NSF grants CCF-0726514, CNS-0715397 and CCF-0937274.
†The work of this author was supported by the NSERC postdoctoral fellowship.
‡The work of this author was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/G023018/1, EP/H018816/1].

DIALM-POMC'10, September 16, 2010, Cambridge, MA, USA.

collisions. This leads to complicated algorithms and analysis even for simple tasks, and makes it prohibitively difficult to study algorithms for complex high-level problems. Moreover, there are a variety of wireless communication models [18, 8, 17], requiring algorithms to be redesigned and reanalyzed for each new model.

Recently, Kuhn et al. [14, 15] proposed a new approach with the goal of combining the advantages of both previous approaches, while avoiding their major problems. Namely, they defined an *Abstract MAC Layer* service that expresses the key guarantees of real MAC layers with respect to local broadcast. This service accepts message transmission requests from nodes and guarantees delivery to nearby nodes within time that depends on the amount of current local contention. The abstract MAC layer is intended to decompose the effort of designing and analyzing wireless network algorithms into two independent and manageable pieces, one that implements the MAC layer over a physical network, and one that uses the MAC layer to solve higher-level problems. Moreover, the abstract MAC layer provides flexibility, in that it allows different implementations of the layer to be combined easily with different high-level algorithms that use the layer. To illustrate the approach, Kuhn et al. analyzed a greedy multi-message global broadcast protocol in terms of the abstract MAC layer.

Kuhn et al. focused on high-level issues of designing and analyzing algorithms over the MAC layer. They did not address in detail the low-level issues of implementing the abstract MAC layer over a physical network nor issues of combining high-level and low-level algorithms. They also did not consider the probabilistic nature of many MAC layer algorithms. Typical algorithms use techniques such as random backoff, which introduce a small probability that abstract MAC assumptions will be violated. To obtain accurate results for higher-level algorithms, one should also take such probabilities into account.

In this paper: We present a case study that shows how one can combine results about high-level protocols based on an abstract MAC layer with results about algorithms that implement an abstract MAC layer over a physical network, and thereby obtain good overall results for the high-level protocols over the physical network. Specifically, we develop and analyze simple greedy protocols for broadcasting single messages and multiple messages throughout a network, using a slot-based physical network model that includes message collisions without collision detection. Each of our protocols is split formally into a high-level broadcast protocol and a low-level contention-management algorithm. We use abstract MAC layers to encapsulate the contention management. We use two different layers: the basic (non-probabilistic) one of [14, 15] and a new probabilistic layer. For contention management, we use a randomized algorithm *DMAC* that is similar to those in [2, 10]. We show that *DMAC* implements the basic abstract MAC layer with high probability, and that it implements the probabilistic layer precisely.

We then combine *DMAC* with a greedy algorithm for single-message global broadcast and analyze the combination twice, using both abstract MAC layers as intermediate layers. Using the basic MAC layer, we prove that the combined algorithm takes time $O(D \log(\frac{n}{\epsilon}) \log \Delta)$ to deliver the message everywhere with probability $1 - \epsilon$, where D is the network diameter, n is the number of nodes, and Δ is the maximum node degree. Using the probabilistic MAC layer, we prove a bound of $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$, matching the best bound previously obtained without such a split [2]. Our combined algorithm is similar to that of [2]; the key difference is that we decompose the algorithm and analysis into two pieces that can be used and understood independently.

We then present an algorithm for multi-message broadcast, obtaining new bounds of $O((D + k'\Delta) \log(\frac{nk}{\epsilon}) \log \Delta)$ using the basic layer and $O((D + k'\Delta \log(\frac{nk}{\epsilon})) \log \Delta)$ using the probabilistic layer, for the time to deliver a message everywhere in the presence of at most k' concurrent messages, with at most k messages overall. If k is polynomial in n, these bounds reduce to simply $O((D + k'\Delta) \log(\frac{n}{\epsilon}) \log \Delta)$ and $O((D+k'\Delta \log(\frac{n}{\epsilon})) \log \Delta)$. Our analysis for multi-message broadcast over the probabilistic layer is not easy; in fact, it seems infeasible without such a decomposition.

Thus, for both broadcast algorithms, we obtain better bounds using the new probabilistic MAC layer than we do using the basic MAC layer. We first considered just the basic layer, and obtained our bounds for broadcast as easy corollaries of results already proved in [14, 15]. However, the bound we obtained for single-message broadcast was not quite as good as the best known bound, which led us to define the probabilistic layer and reanalyze the high-level broadcast algorithms using that layer.

Discussion: The contributions of this paper are: (1) the definition of the new probabilistic abstract MAC layer, (2) the clean decomposition of a single-message broadcast algorithm similar to that of Bar-Yehuda et al. [2] into two pieces, a greedy high-level algorithm and the *DMAC* contention-management algorithm, that can be used and analyzed independently, and (3) the design and analysis of a multi-message broadcast algorithm based on the broadcast algorithm of [14, 15] combined with *DMAC*. This work demonstrates that it is feasible to design and analyze high-level algorithms for collision-prone physical networks using abstract MAC layers.

More evidence for the value of this approach appears in other recent work: Cornejo et al. [4, 5] have developed new Neighbor Discovery algorithms over the basic abstract MAC layer. These enable the construction of a high-level dynamic graph model like the one used in [19] over an abstract MAC layer, which in turn supports the analysis of many already-existing algorithms based on dynamic graphs, in terms of abstract MAC layers and so in terms of physical network models. Dolev et al. [7] have recently developed three new implementations of our probabilistic layer based on physical network models with multiple channels and adversarial interference; by combining these with our high-level broadcast algorithms, they automatically obtain algorithms and bounds for global broadcast for all three models. Khabbazian et al. [12] are currently studying implementations of abstract MAC layers based on network coding techniques like Zig-Zag decoding [9].

Related work: This work relies on [14, 15] for the general idea of decomposing wireless network algorithms using an abstract MAC layer, as well as the basic layer specification and the greedy multi-message global broadcast algorithm. Adler and Scheideler [1] also analyzed high-level wireless network protocols in terms of an abstract MAC layer. They considered the problem of point-to-point message routing, and used a different MAC layer model, which relates message delivery to signal strength.

The problem of single-message global broadcast in an ad hoc radio network was introduced in [2]. That paper contains a randomized algorithm that accomplishes the task in $O((D+$

$\log(n/\epsilon))\log\Delta)$ steps with probability $\geq 1-\epsilon$. Our single-message broadcast algorithm was inspired directly by this algorithm; essentially, we split the algorithm and its analysis into two parts, while retaining the time bound. Subsequently, numerous papers have addressed this problem, e.g., [13, 6] obtain a bound of $O((D+\log(n/\epsilon))\log(n/D))$, which improves upon [2] for dense networks with large diameters.

The problem of multi-message global broadcast has not been widely studied. A randomized algorithm for delivering k messages was given in [3]; it relies on a BFS tree built in a set-up phase prior to the broadcast requests, and routes all messages through the root of the tree. The overall cost is $O((n+(k+D)\log(n/\epsilon))\log\Delta)$, with probability $1-\epsilon$. Our algorithm is faster for cases where $k'\Delta < k+D$. Our algorithm does not require any precomputation and is much simpler (the high-level algorithm is a trivial greedy algorithm) and more robust (the algorithm is symmetric and does not have a single point of failure). The paper [6] contains a randomized algorithm for n simultaneous broadcasts working in time $O(n\log(n/\epsilon)\log n)$ with probability $\geq 1-\epsilon$. This algorithm differs from ours and that of [3] in that it allows intermediate nodes to combine an arbitrary amount of information into a single message, thus reducing high-level contention. In this prior work on broadcast, the issues involving broadcast and contention management are intermingled.

A full version of this work, including fully detailed proofs, appears in [11].

2. MATHEMATICAL PRELIMINARIES

2.1 Graph Theory

Throughout, we fix a (static) connected undirected communication graph $G=(V,E)$. Let $n=|V|$ be the number of nodes in G, and let $\Delta \geq 1$ be the maximum node degree. Fix $\sigma = (\lceil \log(\Delta+1)\rceil)$. Let $dist(i,j)$ denote the distance (the length, in hops, of a shortest path) between nodes i and j. Let D be the diameter of G, that is, the maximum distance between any two nodes in G. If $i \in V$, then let $\Gamma(i)$ be the set of nodes consisting of i and all of its neighbors in G. If $I \subseteq V$, then we define $\Gamma(I) = \bigcup_{i\in I}\Gamma(i)$. For every $i,j \in V$, we fix a shortest path $P_{i,j}$ from i to j in G. We assume that these paths are *consistent*, that is, for every $i,j,i',j' \in V$, if nodes i' and j' appear, in that order, on path $P_{i,j}$, then path $P_{i',j'}$ is a subpath of $P_{i,j}$.

2.2 Probability

We formalize our results in terms of *Probabilistic Timed I/O Automata (PTIOAs)*, as defined by Mitra [16]. PTIOAs include mechanisms (local schedulers and task schedulers) to resolve nondeterminism. We consider probabilistic executions of systems modeled as PTIOAs. We analyze the probabilities of events, which are sets of time-unbounded executions. These probabilities are taken with respect to the probability distribution that arises by considering the entire probabilistic execution, starting from the initial system state. We also consider probabilities with respect to a "cone" in the full probabilistic execution following a particular closed finite execution β. More precisely, we consider the conditional probability distribution on the set A_β of time-unbounded executions that extend β. We denote this probability distribution by $\Pr_\beta$.

The following lemma encapsulates a Chernoff bound analysis that we use twice in the paper.

LEMMA 2.1. *Let $Y_q, q=1,\ldots$ be a collection of independent $\{0,1\}$-valued random variables, each equal to 1 with probability $p>0$. Let d and τ be nonnegative reals, $d \geq 1$. Let $r = \left\lfloor \frac{1}{p}(3d+2\tau)\right\rfloor$. Then $\Pr\left(\sum_{q=1}^{r} Y_q < d\right) \leq e^{-\tau}$.*

3. THE PHYSICAL MODEL

We assume a collection of n probabilistic processes. We assume that time is divided into *slots*, each of real-time duration 1. Processes have synchronized clocks, and so can detect when each slot begins and ends. Processes communicate only on slot boundaries. All processes awaken at the same time 0, which is the beginning of slot 1. We assume that each process has both transmitter and receiver hardware. The receivers operate at every slot, and processes decide when to transmit.

We assume that the n processes reside at the nodes of communication graph G, one per node. Processes know n and Δ, but nothing else about the graph. We assume a physical network, *Net*, with collisions but no collision detection. When a process transmits in some slot, its message *reaches* exactly itself and all G-neighbors. What process j *receives* is defined as follows: If j is reached by its own message, then it receives just its own message, regardless of whether it is reached by any other messages. Otherwise, if j is reached by exactly one message (from another process), then it receives that message, and if it is reached by no messages or by two or more messages, it receives silence, represented as $\perp$. Thus, processes cannot distinguish collisions from silence.

4. ABSTRACT MAC LAYERS

We specify the two abstract MAC layers, a special case of the basic layer of [14, 15], and a new probabilistic layer. Our layers are defined for a single, static, undirected communication graph $G=(V,E)$ with maximum node degree Δ. Both specifications are implicitly parameterized by three positive reals, f_{rcv}, f_{ack}, and f_{prog}. These bound delays for a specific message to arrive at a particular receiver, for an acknowledgement to arrive at a sender, and for *some* message from among many competing messages to arrive at a receiver, respectively.* In typical MAC implementations, f_{prog} will be notably smaller than f_{rcv} and f_{ack}, as the time for delivering *some* message to a receiver is typically substantially shorter than the upper bounds for delivering *every* message and for getting an acknowledgement. Both specifications also use a (small) nonnegative real parameter t_{abort}, which bounds the amount of time after a sender aborts a sending attempt when the message could still arrive at some receiver. Both specifications present an interface to higher layers with inputs $bcast(m)_i$ and $abort(m)_i$ and outputs $rcv(m)_i$ and $ack(m)_i$, for every m in a given message alphabet M and every $i \in V$.

We model a MAC layer formally as a PTIOA *Mac*. To implement either of our specifications, *Mac* must guarantee several conditions whenever it is composed with any probabilistic environment *Env* and the physical network *Net* (also modeled as PTIOAs). The composition $Mac \| Env \| Net$ (also a PTIOA) yields a unique probabilistic execution, that is, a unique probability distribution on executions. To define the guarantees of the MAC layers, we assume some "well-formedness" constraints on the environment *Env*: An execution α of $Mac \| Env \| Net$ is *well-formed* if (a) it contains at

*Since our bounds do not depend on the actual contention, but only on maximum node degree, we do not express these bounds as functions of the contention as in [14, 15].

most one *bcast* event for each $m \in M$ (all messages are unique), (b) any $abort(m)_i$ event in α 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 in α have an intervening ack_i or $abort_i$.

4.1 The Basic Abstract MAC Layer

Our *Basic Abstract MAC Layer* specifies worst-case bounds for receive, acknowledgement, and progress delays. The specification says that the *Mac* automaton guarantees the following, 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, where $i \neq j$, and that also maps each $ack(m)_i$ and $abort(m)_i$ to a preceding $bcast(m)_i$. This must satisfy:

1. *Receive restrictions:* If a $bcast(m)_i$ 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 acknowledgements:* No $ack(m)_i$ caused by π precedes π'.

2. *Acknowledgement restrictions:* If $bcast(m)_i$ event π causes $ack(m)_i$ event π', then (a) *Guaranteed communication:* If $(i,j) \in E$ then a $rcv(m)_j$ caused by π precedes π'. (b) *No duplicate acknowledgements:* No other $ack(m)_i$ caused by π precedes π'. (c) *No acknowledgements after aborts:* No $abort(m)_i$ caused by π precedes π.

3. *Temination:* Every $bcast(m)_i$ causes either an $ack(m)_i$ or an $abort(m)_i$.

A *message instance* in α is a matched pair of *bcast*/*ack* or *bcast*/*abort* events. The specification also says that the *Mac* automaton guarantees the following three upper bounds on message delays. Here, f_{rcv} bounds the time for a specific message to arrive at a particular receiver, f_{ack} bounds the time for an acknowledgement to arrive at a sender, and f_{prog} bounds the time for *some* message to arrive at a receiver.

1. *Receive delay bound:* If a $bcast(m)_i$ event π causes a $rcv(m)_j$ event π', then the time between π and π' is at most f_{rcv}. Furthermore, if there exists an $abort(m)_i$ event π'' such that π causes π'', then π' does not occur more than t_{abort} time after π''.

2. *Acknowledgement delay bound:* If a $bcast(m)_i$ event π causes an $ack(m)_j$ event π', then the time between π and π' is at most f_{ack}.

3. *Progress bound:* If α' is a closed execution fragment within α and j is any node, then it is not the case that all three of the following conditions hold: (a) The duration for α' is strictly greater than f_{prog}. (b) At least one message instance from a neighbor of j completely contains α'. (c) No rcv_j event of a message instance that overlaps α' occurs by the end of α'.

4.2 The Probabilistic Abstract MAC Layer

Our *Probabilistic Abstract MAC Layer* specifies probabilistic bounds for the three kinds of delays. In addition to the four parameters above, this specification uses parameters ϵ_{prog}, ϵ_{rcv}, and ϵ_{ack}, representing error probabilities for attaining the delay bounds. This specification says that, for every well-formed execution α of $Mac \| Env \| Net$, there exists a *cause* function as before, satisfying the following non-probabilistic properties defined in Section 4.1: all the *Receive restrictions*, *No duplicate acknowledgements*, and *No acknowledgements after aborts*. Moreover, no *rcv* happens more than t_{abort} time after a corresponding *abort*. Note that we have omitted the *Guaranteed communication* and *Termination* properties from this list; we fold these into the acknowledgement delay bound, below.

The specification also says that the *Mac* automaton must guarantee the following probabilistic upper bounds on message delays. If β is a closed execution, then we say that a *bcast* event in β is *active at the end of* β provided that it is not terminated with an *ack* or *abort* in β. Assume $i, j \in V$, and t is a nonnegative real.

1. *Receive delay bound:* Let j be a neighbor of i. 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 $rcv(m)_j$ occurs by time $t + f_{rcv}$.
   If $\Pr_\beta(A) > 0$, then $\Pr_\beta(B|A) \geq 1 - \epsilon_{rcv}$.

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

3. *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. 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 message whose *bcast* occurs after β.
   If $\Pr_\beta(A) > 0$, then $\Pr_\beta(B|A) \geq 1 - \epsilon_{prog}$.

The progress bound says that, if a nonempty set of j's neighbors have active *bcast*s at some point, and none of these messages has yet been received by j, then with probability at least $1 - \epsilon_{prog}$, within time f_{prog}, either j receives one of these or something newer, or else all of these end with *acks*. This is all conditioned on the absence of aborts.

5. CONTENTION MANAGEMENT

We implement our abstract MAC layers using a contention management algorithm *DMAC*. *DMAC* uses a probabilistic retransmission strategy similar to the Decay strategy of [2] and the Probability-Increase strategy of [10]. We prove two results: Theorem 5.7 says that *DMAC* implements the probabilistic abstract MAC layer (exactly), and Theorem 5.8 says that it implements the basic abstract MAC layer with high probability.

5.1 The DMAC Algorithm

Our retransmission strategy differs slightly from the one in [2] in that the processes successively increase their transmission probabilities in a Decay phase rather than decrease them. Also, our processes choose randomly whether to transmit in each individual slot, whereas in [2], they choose randomly whether to drop out of the current Decay phase. We give a lower bound on the success probability for our algorithm. The algorithm uses knowledge of Δ, the maximum degree in G.

Decay: This algorithm runs for exactly $\sigma = \lceil \log(\Delta+1) \rceil$ slots. A set I of processes, $|I| \leq \Delta$, plus another distinguished process j, participate. We assume that at least one process in I participates in all slots. Other processes in I, and also j, may participate in some slots, but once they stop participating, they do not participate in later slots. At each slot $s = 1, \ldots, \sigma$, each participating process transmits with probability p_s, where $p_\sigma = \frac{1}{2}, p_{\sigma-1} = \frac{1}{2^2}, \ldots, p_{\sigma-s} = \frac{1}{2^{s+1}}, \ldots, p_1 = \frac{1}{2^\sigma}$.

LEMMA 5.1. *In Decay, with probability at least $\frac{1}{8}$, at some slot, some process in I transmits alone (that is, without any other process in I transmitting and without j transmitting).*

PROOF. We first show that, at some slot s, the number of participants c_s satisfies $\frac{1}{2c_s} \leq p_s \leq \frac{1}{c_s}$. Then, using a case analysis based on whether or not j participates in this slot s, we show that, in either case, the probability that some process in I transmits alone at slot s is at least $\frac{1}{8}$. □

Our MAC algorithm is $DMAC(\phi)$, where ϕ indicates the number of Decay phases that are executed.

***DMAC*(ϕ):** We group slots into *Decay phases*, each consisting of σ slots. Each MAC layer process i that receives a message from its environment, via a $bcast(m)_i$ event, starts executing Decay with message m (and a unique message identifier) at the beginning of the next Decay phase. Process i executes exactly ϕ Decay phases, and then outputs $ack(m)_i$ at the end of the final phase. However, if process i receives an $abort(m)_i$ from the environment before it performs $ack(m)_i$, it performs no further transmission on behalf of message m and does not perform $ack(m)_i$.
Meanwhile, process i tries to receive, in every slot. When it receives any message m' from a neighbor (not from itself) for the first time on the physical network, it delivers that to its environment with a $rcv(m')_i$ event, at a real time before the ending time of the slot.

Note that, in $DMAC(\phi)$, no process starts participating in a Decay phase part-way through the phase, but it may stop participating at any time as a result of an *abort*. Define $DMAC(\phi)$ to be the composition of $DMAC(\phi)_i$ processes for all i. We prove five lemmas giving properties of $DMAC(\phi)$. Lemma 5.2 asserts that $DMAC(\phi)$ satisfies all of the non-probabilistic guarantees, and Lemma 5.3 asserts an absolute bound on acknowledgement time.

LEMMA 5.2. *In every time-unbounded execution, the* Proximity, No duplicate receives, No receives after acknowledgements, No duplicate acknowledgements, *and* No acknowledgements after aborts *conditions are satisfied. Also, no rcv happens more than time 1 after a corresponding abort.*

LEMMA 5.3. *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 after exactly ϕ Decay phases, starting with the next phase that begins after the $bcast(m)_i$.*

For the remaining three lemmas, we fix any probabilistic environment Env, and consider the unique probabilistic execution of $DMAC(\phi) \| Env \| Net$. The lemmas give probabilistic delay bounds for progress and receives, and a probabilistic guarantee that acknowledgements are preceded by receives. In these lemmas, ϵ is a real, $0 < \epsilon \leq 1$; ϕ, the parameter for $DMAC(\phi)$, is equal to $\lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$; and g and h are positive integers. The proofs use standard probabilistic arguments and Lemma 5.1.

LEMMA 5.4. *Let $i, j \in V$, i a neighbor of j. Let β be a closed execution that ends with $bcast(m)_i$ at time t, where $(g-1)\sigma \leq t < g\sigma$.*
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 Decay phase $g + \phi$, a $rcv(m)_j$ occurs.
If $\Pr_\beta(A) > 0$, then $\Pr_\beta(\bar{A} \cup B) \geq \Pr_\beta(B|A) \geq 1 - \epsilon$.

LEMMA 5.5. *Let $i \in V$. Let β be any closed prefix of a time-unbounded execution that ends with $bcast(m)_i$ at time t, where $(g-1)\sigma \leq t < g\sigma$.*
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 Decay phase $g + \phi$, $ack(m)_i$ occurs and is preceded by $rcv(m)_j$ for every neighbor j of i.
If $\Pr_\beta(A) > 0$, then $\Pr_\beta(\bar{A} \cup B) \geq \Pr_\beta(B|A) \geq 1 - \epsilon\Delta$.

LEMMA 5.6. *Let $j \in V$. Let β be a closed execution that ends at time t, where $(g-1)\sigma \leq t < g\sigma$. 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 $i \in I$.*
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 $g + h$, at least one of the following occurs: a $rcv(m_i)_j$ for some $i \in I$, or a rcv_j for some message whose bcast occurs after β, and C, the executions in which, by the end of Decay phase $g + h$, $ack(m_i)_i$ occurs for every $i \in I$.
If $\Pr_\beta(A) > 0$, then $\Pr_\beta(\bar{A} \cup B \cup C) \geq \Pr_\beta(B \cup C|A) \geq 1 - (\frac{7}{8})^h$.

5.2 Implementing the Probabilistic Layer

First, we show that $DMAC(\phi)$ implements the probabilistic layer, for a particular choice of ϕ. Here we fix constants: ϵ, a real, $0 < \epsilon \leq 1$; h, a positive integer (the number of Decay phases for the progress bound); and $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$ (the number of Decay phases for the receive and acknowledgement bounds). We define the parameters for the probabilistic layer: $f_{rcv} = f_{ack} = (\phi + 1)\sigma$; $f_{prog} = (h+1)\sigma$; $\epsilon_{rcv} = \epsilon$; $\epsilon_{ack} = \epsilon\Delta$; $\epsilon_{prog} = (\frac{7}{8})^h$; and $t_{abort} = 1$. Using Lemmas 5.2-5.6, we obtain:

THEOREM 5.7. *$DMAC(\phi)$ implements the probabilistic layer with parameters $f_{rcv} = f_{ack} = O(\Delta \log(\frac{1}{\epsilon}) \log \Delta)$, $f_{prog} = O(h \log \Delta)$, $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, $\epsilon_{prog} = (\frac{7}{8})^h$, and $t_{abort} = O(1)$.*

5.3 Implementing the Basic Layer

Now we prove that, with probability $\geq 1-\epsilon$, $DMAC(\phi)$ implements the basic layer, for certain values of ϵ and ϕ. Our theorem assumes that, in any execution, the environment *Env* submits at most b *bcast*s, for some fixed positive integer b, so the total number of external MAC layer events (*bcast*, *ack*, *abort*, and *rcv*) is at most $b(\Delta+2)$. Here we fix: ϵ, a real, $0<\epsilon\leq 1$; $a = b(\Delta+2)$; $\epsilon_1 = \frac{\epsilon}{2a}$ (a bound on the errors for some individual properties); $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1})\rceil$ (the number of Decay phases for the acknowledgement bound); and $h = \log_{8/7}(1/\epsilon_1)$. We also define the parameters for the basic layer: $f_{rcv} = f_{ack} = (\phi+1)$ *sigma*; $f_{prog} = (\lceil h\rceil+1)\sigma$; and $t_{abort} = 1$. In the following result, the probabilistic claims are with respect to the unique probabilistic execution of the system $DMAC(\phi)\|Env\|Net$, where *Env* is a probabilistic environment that submits at most b *bcast*s.

THEOREM 5.8. *With probability at least $1-\epsilon$, the execution satisfies all the properties of the basic layer, with f_{rcv}, f_{ack}, f_{prog}, and t_{abort} as defined above, and thus, with $f_{rcv} = f_{ack} = O(\Delta\log(\frac{\Delta b}{\epsilon})\log\Delta)$, $f_{prog} = O(\log(\frac{\Delta b}{\epsilon})\log\Delta)$, and $t_{abort} = O(1)$.*

PROOF. Theorem 5.7 implies that the algorithm satisfies the non-probabilistic properties. By Lemma 5.3, for every $bcast_i$ event that is not terminated with an *abort*, a corresponding ack_i occurs within ϕ Decay phases, and so by time $f_{ack} = (\phi+1)\sigma$. Thus, if the implementation fails for an execution, it must be because of either an *ack violation*, meaning that some *ack* event is not preceded by all required *rcv* events, or a *progress violation*, meaning that the progress delay bound is violated somewhere. We show that the probability of each of these types of violations is at most $\frac{\epsilon}{2}$. These proofs involve applying Lemma 5.5 for individual *bcast* events and Lemma 5.6 for individual external MAC layer events, and using union bounds. □

6. GLOBAL BROADCAST

In the *multi-message broadcast (MMB)* problem, messages arrive from the environment at arbitrary times, at arbitrary locations, via $arrive(m)_i$ inputs. The algorithm is supposed to deliver all messages to all locations, using $deliver(m)_i$ outputs. The *single-message broadcast (SMB)* problem is essentially the special case of the MMB problem for a single message originating at a single (known) location i_0.

Our broadcast algorithms are based on the Basic Multi-Message Broadcast (BMMB) algorithm of [14, 15]. This algorithm is intended to be combined with a (basic or probabilistic) MAC layer.

The Basic Multi-Message Broadcast (BMMB) Protocol: Every process i maintains a FIFO queue named *bcastq* and a set named *rcvd*. Both are initially empty. If process i is not currently sending a message on the MAC layer and its *bcastq* is not empty, it sends the message at the head of the queue on the MAC layer (disambiguated with identifier i and sequence number) using a *bcast* output. If i receives a message from the environment via an $arrive(m)_i$ input, it immediately delivers the message m to the environment using a $deliver(m)_i$ output, and adds m to the back of *bcastq* and to the *rcvd* set. If i receives a message m from the MAC layer via a $rcv(m)_i$ input, it first checks *rcvd*. If $m \in rcvd$ it discards it. Else, i immediately performs a $deliver(m)_i$ output and adds m to *bcastq* and *rcvd*.

The Basic Single-Message Broadcast (BSMB) Protocol: This is just *BMMB* specialized to one message, and modified so that the message starts in the state of a designated initial node i_0.

We combine these with our *DMAC* implementation of the MAC layer, parameterizing the combined algorithms with the number ϕ of Decay phases. Namely, *BSMB*-Decay(ϕ) consists of *BSMB* composed with $DMAC(\phi)$; this combination is similar to the global broadcast algorithm in [2]. *BMMB*-Decay(ϕ) consists of *BMMB* and $DMAC(\phi)$.

7. ANALYSIS OF THE SINGLE-MESSAGE BROADCAST ALGORITHM

We analyze *BSMB*-Decay using both the basic and probabilistic MAC layers, by combining results from Section 5 with higher-level analysis of the global broadcast algorithm. Our results, Theorems 7.1 and 7.9, take the form of assertions that, with probability at least $1-\epsilon$, for an arbitrary ϵ, $0<\epsilon\leq 1$, the message is delivered everywhere within time that is expressed as a function of ϵ and graph parameters.

The analysis using the basic layer is very simple, because it uses previous high-level analysis results without modification. However, it yields a slightly worse bound than the one obtained by Bar-Yehuda et al. for the intermingled algorithm [2]. The analysis using the probabilistic layer yields the same bounds as in [2], but the high-level analysis of *BSMB* over the MAC layer must be redone, and is not easy. The ideas in this analysis are derived from those in the portion of the analysis of [2] that deals with the high-level algorithm, with a little extra complication due to asynchrony. Our main accomplishment here is that we have decomposed the algorithm and its analysis into two independent pieces.

7.1 Basic Abstract MAC Layer

Here we use our basic MAC layer to prove an upper bound of $O(D\log(\frac{n}{\epsilon})\log\Delta)$ on the time to deliver the message everywhere with probability at least $1-\epsilon$. To define ϕ (the number of Decay phases), in the theorem statement, we define constants: $b = n$ (a bound on the number of *bcast* events—here, the single message gets *bcast* at most once by each node); $a = n(\Delta+2)$ (a bound on the total number of external MAC layer events); $\epsilon_1 = \frac{\epsilon}{2a}$; and $\phi = \lceil 8\Delta\ln(\frac{\Delta}{\epsilon_1})\rceil$.

THEOREM 7.1. *With probability at least $1-\epsilon$, BSMB-Decay(ϕ) guarantees that deliver events occur at all nodes $\neq i_0$ by time $O(D\log(\frac{n}{\epsilon})\log\Delta)$.*

PROOF. Theorem 3.2 of [15] implies that the message is received everywhere within time $O(Df_{prog})$. Based on the constants in the first part of Section 5, and using the assumption that $\sigma = (\lceil\log(\Delta+1)\rceil)$, we substitute $f_{prog} = O(h\log\Delta), h = O(\log(\frac{1}{\epsilon_1})), \epsilon_1 = \frac{\epsilon}{2a}$, and $a = O(n\Delta)$, to obtain a bound of the form $O(D\log(\frac{n}{\epsilon})\log\Delta)$. This means that, if the algorithm ran with a basic abstract MAC layer with f_{prog} as above, it would, in every execution, deliver the message everywhere by the indicated time. Since Theorem 5.8 implies that the MAC layer achieves the progress bound f_{prog} with probability at least $1-\epsilon$, the entire system achieves the required message delivery bound with probability at least $1-\epsilon$. □

7.2 Probabilistic Abstract MAC Layer

Now we use our probabilistic MAC layer to improve the bound of Section 7.1 to $O((D+\log(\frac{n}{\epsilon}))\log\Delta)$, the same

bound as in [2]. We first assume a probabilistic layer with parameters f_{prog}, f_{ack}, ϵ_{prog}, and ϵ_{ack} and analyze the complexity of *BSMB* in terms of these parameters. Then we replace the abstract layer with *DMAC* and combine our bounds for *DMAC* with our bounds for *BSMB* to obtain Theorem 7.9. We begin by identifying "nice" broadcasts, which result in timely acknowledgements preceded by all receives: We bound the probability that an execution is not nice:

DEFINITION 7.2 (NICE BROADCASTS AND EXECUTIONS). *Suppose a* $bcast(m)_i$ *event* π *occurs at time* t_0 *in execution* α. *Then* π *is* nice *if* $ack(m)_i$ *occurs by time* $t_0 + f_{ack}$ *and is preceded by a* $rcv(m)_j$ *for every neighbor* j *of* i. *Execution* α *is* nice *if all bcast events in* α *are nice. Let* N *denote the set of all nice executions.*

LEMMA 7.3. *In a system of the form* $Mac \| Env \| Net$, *where Mac implements the probabilistic abstract MAC layer with acknowledgement parameters* f_{ack} *and* ϵ_{ack}, *and Env is an environment for the MAC layer that submits at most b bcasts in any execution and never submits an abort,* $\Pr(\bar{N}) \leq b\epsilon_{ack}$.

PROOF. Using the definition of f_{ack}, we argue that the probability of a violation involving each *bcast* event is at most ϵ_{ack}. The result follows from a union bound. □

Successful progress for the message is captured formally in the following "Progress Condition", which is parameterized by a nonnegative real τ (a time duration). We also include a (small) parameter δ, because of a race condition that arises from the combination of probability and asynchrony (see below).

DEFINITION 7.4 ($PC_j^\delta(\tau)$). *Assume that* $j \in V - \{i_0\}$ *and that* δ *and* τ *are nonnegative reals. Define constants* $\gamma_1 = \frac{3}{1-\epsilon_{prog}}$ *and* $\gamma_2 = \frac{2}{1-\epsilon_{prog}}$. *We say that* $\alpha \in PC_j^\delta(\tau)$ *if a* rcv_j *event occurs in* α *by time* $(\gamma_1 dist(i_0, j) + \gamma_2\tau)(f_{prog} + \delta)$. *Let* $PC^\delta(\tau) = \bigcap_j PC_j^\delta(\tau)$. *Let* $PC_j(\tau)$ *and* $PC(\tau)$ *denote* $PC_j^0(\tau)$ *and* $PC^0(\tau)$, *respectively.*

In the following results, our probabilistic statements are with respect to the system $BSMB \| Mac \| Env \| Net$, where *Mac* is an arbitrary implementation of the abstract probabilistic MAC layer with parameters f_{prog}, f_{ack}, ϵ_{prog}, and ϵ_{ack}, and *Env* is some probabilistic environment. Most of the work in our analysis is devoted to proving the following lemma, which lower-bounds the probability of the progress condition $PC_j^\delta(\tau)$. It works for any positive value of δ, no matter how small—any such δ suffices to handle the race conditions.

LEMMA 7.5. *Let* $j \in V - \{i_0\}$. *Let* δ *be a positive real. Then* $\Pr(PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}$.

PROOF. See Lemma 6.6 of [11]. We consider the distinguished shortest path $P_{i_0,j}$ from i_0 to j. For every q, we define time $t_q = q(f_{prog} + \delta)$. We define a random variable $Dist_q$ to capture the maximum progress made by the message along the path by time t_q, and a Boolean random variable X_q to indicate whether progress is made between times t_q and t_{q+1}; X_q is essentially $min(1, Dist_{q+1} - Dist_q)$. We prove the key fact that, for any finite execution β that ends at time $t_q + \delta$, the probability that $X_q = 1$, that is, that progress is made between times t_q and t_{q+1}, conditioned on an execution being an extension of β, is at least $1 - \epsilon_{prog}$. Combining this result for all such β yields that the probability that $X_q = 1$, conditioned on any values of $X_0, X_1, \ldots, X_{q-1}$, is at least $1 - \epsilon_{prog}$. Also, the probability that $X_0 = 1$ is at least $1 - \epsilon_{prog}$. We then apply Lemma 2.1 (where the Y_q are 1 with probability exactly $1 - \epsilon$) to obtain the final bound. □

The δ is used in the proof to guarantee that the values of random variables $Dist_0, Dist_1, \ldots, Dist_q$ and $X_0, X_1, \ldots, X_{q-1}$ are really determined by the prefix β—this does not follow automatically. Nevertheless, we can remove the δ from the statement of the lemma:

LEMMA 7.6. *Let* $j \in V - \{i_0\}$. *Then* $\Pr(PC_j(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}$.

PROOF. See Lemma 6.7 of [11]. The key fact is that Lemma 7.5 holds for every $\delta > 0$. □

Standard probabilistic arguments, including union bounds, then yield a lower bound on the set of executions that are nice, and also satisfy the progress condition for every node j:

LEMMA 7.7. $\Pr(PC(\tau) \cap N) \geq 1 - ne^{-\tau} - \Pr(\bar{N})$.

We combine Lemma 7.7 with Lemma 7.3 (our upper bound on the probability of $\bar{N}$), and instantiate τ as $\ln(\frac{n}{\epsilon})$, to obtain our bound for *BSMB* over the probabilistic MAC layer:

THEOREM 7.8. *BSMB guarantees that, with probability at least* $1 - \epsilon - n\epsilon_{ack}$, *deliver events occur at all nodes* $\neq i_0$ *by time* $(\gamma_1 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}$.

Finally, we combine the bound for *BSMB* in terms of the probabilistic layer with our bound for *DMAC* to obtain a bound for *BSMB*-Decay. Our probabilistic statement is for the *BSMB*-Decay$(\phi) \| Env \| Net$, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$, and *Env* is some probabilistic environment.

THEOREM 7.9. *Let* $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$. *Then with probability at least* $1 - \epsilon$, *BSMB*-Decay(ϕ) *guarantees that deliver events occur at all nodes* $\neq i_0$ *by time* $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$.

PROOF. Choose $\epsilon_{ack} = \frac{\epsilon}{2n}$. Theorem 7.8, applied with ϵ in that theorem instantiated as our $\frac{\epsilon}{2}$, implies that, with probability at least $1 - \frac{\epsilon}{2} - n\epsilon_{ack} \geq 1 - \epsilon$, *rcv* events occur at all nodes $\neq i_0$ by time $(\gamma_1 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}$. Using the definitions of parameters for the implementation of the probabilistic layer, in Section 5, we may assume that $\epsilon_{prog} \leq \frac{7}{8}$ to make this $O((D + \log(\frac{n}{\epsilon}))f_{prog})$. Again using the parameter definitions, we substitute $f_{prog} = O(\log \Delta)$ into the expression, to get a bound of $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$. □

8. ANALYSIS OF THE MULTI-MESSAGE BROADCAST ALGORITHM

Now we analyze *BMMB*-Decay, using both the basic and probabilistic MAC layers, by combining results from Section 5 with higher-level analysis of the global broadcast algorithm. Our results, Theorems 8.2 and 8.12, assert probabilistic upper bounds on the time for delivering any particular message to all nodes in the network, in the presence of a limited number of concurrent messages. We assume a bound k on the number of messages that arrive from the environment during the entire execution. Again, the analysis using the basic layer is simple, while the analysis using the probabilistic layer is more difficult and yields a better bound. The latter analysis is new; it uses ideas from the

analysis in Section 7.2, plus new ideas to cope with the online arrival of messages. It also uses a path decomposition trick to achieve the best bound. We begin by defining the set $K(m)$ of messages concurrent with a given message m:

DEFINITION 8.1 ($K(m)$). *Suppose α is an execution in N and $m \in M$ is a message for which an arrive(m) event occurs in α. Let clear(m) be the final ack(m) event in α. Define $K(m)$ to be the set of messages $m' \in M$ such that an arrive(m') event precedes the clear(m) event and the clear(m') event follows the arrive(m) event. That is, $K(m)$ is the set of messages whose processing overlaps the interval between the arrive(m) and clear(m) events.*

8.1 Basic Abstract MAC Layer

Here we use our basic MAC layer to prove an upper bound of $O((D + k'\Delta)\log(\frac{nk}{\epsilon})\log\Delta)$ on the time to deliver any particular message everywhere with probability at least $1 - \epsilon$, in the presence of at most k' concurrent messages and with at most k messages overall. If k is polynomial in n, the bound reduces to $O((D + k'\Delta)\log(\frac{n}{\epsilon})\log\Delta)$. We define constants: $b = kn$ (a bound on the number of *bcast* events—each of the k messages gets *bcast* at most once by each node); $a = kn(\Delta + 2)$ (a bound on the total number of external MAC layer events); $\epsilon_1 = \frac{\epsilon}{2a}$; and $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1})\rceil$.

THEOREM 8.2. *Let $m \in M$. Then BMMB-Decay(ϕ) guarantees that, with probability at least $1 - \epsilon$, the following property holds of the generated execution α. Suppose an arrive(m) event occurs in α. Let k' be a positive integer such that $|K(m)| \le k'$. Then deliver(m) events occur at all processes in α within time $O((D+k'\Delta)\log(\frac{nk}{\epsilon})\log\Delta)$ of the time of the arrive(m) event.*

PROOF. Theorem 3.2 of [15] implies that the message is received everywhere within time $(D + 2k' - 1)f_{prog} + (k' - 1)f_{ack}$, which is $O((D + k'))f_{prog} + (k' - 1)f_{ack}$. Based on the constants defined in Section 5, we substitute $f_{prog} = O(\log(\frac{1}{\epsilon_1})\log\Delta)$, $f_{ack} = O(\Delta\log(\frac{\Delta}{\epsilon_1})\log\Delta)$, $\epsilon_1 = \frac{\epsilon}{2a}$, and $a = O(kn\Delta)$, to obtain a bound of the form $O\big((D + k'\Delta)\cdot \log(\frac{nk}{\epsilon})\log\Delta\big)$. Theorem 5.8 implies that the MAC layer achieves f_{prog} and f_{ack} with probability $\ge 1-\epsilon$, so the entire system achieves the required message delivery bounds with probability $\ge 1 - \epsilon$. □

8.2 Probabilistic Abstract MAC Layer

Now we use our probabilistic layer to improve the bound of Section 8.1 to $O((D + k'\Delta\log(\frac{nk}{\epsilon}))\log\Delta)$. We first assume a probabilistic layer with parameters f_{prog}, f_{ack}, ϵ_{prog}, and ϵ_{ack} and analyze the complexity of *BMMB* in terms of these parameters. Then we replace the abstract layer with *DMAC* and combine the bounds for *DMAC* and *BMMB* to obtain Theorem 8.12.

For our analysis of *BMMB* over the MAC layer, we first redefine the progress condition PC. Then we prove a non-probabilistic bound on the message delivery time in executions that are "well-behaved", in the sense that they satisfy the new PC, and also are "nice" as defined in Section 7.2. Then we bound the probability that an execution is well-behaved and use this to infer our probabilistic bound on message delivery time. We begin by defining $C_i(t)$, the set of messages whose processing is completed at node i by time t, and also redefining PC. Except in Lemma 8.5, τ is any nonnegative real.

DEFINITION 8.3. *For any $i \in V$, nonnegative real t and execution α, let $C_i^\alpha(t)$ be the set of messages m such that ack$(m)_i$ occurs by time t in α. For any $I \subseteq V$, nonnegative real t and execution α, let $C_I^\alpha(t) = \bigcap_{i\in I} C_i^\alpha(t)$, that is, the set of messages m such that ack$(m)_i$ occurs by time t for every $i \in I$.*

We define a $get(m)_j$ event to be the first event by which node j receives message m; this may be either an *arrive* event by which m arrives from the environment, or a *rcv* event by which m is received from the MAC layer.

DEFINITION 8.4 (PROGRESS CONDITION $PC_{i,j}(\tau)$). *Let $i, j \in V$, $i \ne j$ be two processes. Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$. We say that the progress condition, $PC_{i,j}(\tau)$, holds for an execution α (i.e., $\alpha \in PC_{i,j}(\tau)$) if for every nonnegative real t, the following holds: If a get$(m)_i$ event for some message $m \notin C_{\Gamma(I)}^\alpha(t)$ occurs in α by time t, then a get$(m')_j$ event for some message $m' \notin C_{\Gamma(I)}^\alpha(t)$ occurs by time $t + (\gamma_1 d + \gamma_2\tau)f_{prog}$. Let $PC(\tau) = \bigcap_{i,j,i\ne j} PC_{i,j}(\tau)$.*

8.2.1 Message Delivery for Well-Behaved Executions

Here we prove a non-probabilistic upper bound on message delivery time in well-behaved executions, that is, executions that are in $PC(\tau)\cap N$. This says that, if a message m arrives at a node i from the environment at time t_0, j is any node, and l is any number, then either m reaches j within a certain time that depends on $dist(i, j)$ and l, or else l new messages reach j in that time. The proof is the most challenging one in the paper.

LEMMA 8.5. *Let $\tau, l \ge 1$ and $d \ge 0$ be integers. Define*

$$t_{d,l} := t_0 + ((\gamma_1 + \gamma_2)d + ((\gamma_1 + 2\gamma_2)\tau + \gamma_1 + \gamma_2)l)\, f_{prog} + (l - 1)f_{ack}.$$

Let α be an execution in $PC(\tau)\cap N$. Assume that arrive$(m)_i$ occurs at time t_0 in α. Let $M' \subseteq M$ be the set of messages m' for which arrive$(m)_i$ precedes clear(m') in α. Let $j \in V$, $dist(i, j) = d$. Then for every integer $l \ge 1$, at least one of the following two statements is true:

1. *A get$(m)_j$ event occurs by time $t_{d,l}$ and ack$(m)_j$ occurs by time $t_{d,l} + f_{ack}$.*
2. *There exists a set $M'' \subseteq M'$, $|M''| = l$, such that for every $m' \in M''$, get$(m')_j$ occurs by time $t_{d,l}$ and ack$(m')_j$ occurs by time $t_{d,l} + f_{ack}$.*

PROOF. See Lemma 7.10 of [11]. We prove the lemma by induction on l. For the base case, $l = 1$, we consider subcases based on whether $d = 0$ or $d > 0$. For $d > 0$, we use the fact that $\alpha \in PC_{i,j}(\tau)$ to yield a message with the needed properties. For the inductive step, we assume the lemma for $l - 1$ and all values of d and prove the claim for l. For this, we proceed by induction on d. The base case, $d = 0$, is straightforward. For the inductive step for d, we assume the lemma for l and all smaller values of d. Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$ and let $I = \{i_1, \ldots, i_d\}$. Assume that Statement 1 is false for j and l, i.e., it is not the case that $get(m)_j$ occurs by time $t_{d.l}$ and $ack(m)_j$ occurs by time $t_{d.l} + f_{ack}$. We show that Statement 2 must be true for j and l. By inductive hypothesis, Statement 2 is true for j and $l-1$. That is, there exists $M'' \subseteq M'$, $|M''| = l-1$, such that, for every $m' \in M''$, $get(m')_j$ occurs by time $t_{d,l-1}$ and

$ack(m')_j$ occurs by time $t_{d,l-1} + f_{ack} < t_{d,l}$. Fix this set M'' for the rest of the proof.

After dispensing with easy cases, we are left with the case where $M'' \subseteq C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack})$ for every neighbor j' of j, i.e., all messages in M'' have been completely processed at all neighbors of j. For any integer e, $0 \leq e \leq d-1$, let $I_e = \{i_{e+1}, \ldots, i_d\}$. Let e' be the smallest integer, $0 \leq e' \leq d-1$, such that

$$M'' \subseteq \bigcap_{j' \in \Gamma(I_{e'})} C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack}). \quad (1)$$

We know e' exists because (1) holds for $e' = d-1$. For this e', we show that there exists $m' \in M' - M''$ such that $get(m')_{i_{e'}}$ occurs by time $t_{e'+1,l-1} + f_{ack} < t_{e'+\tau+1,l-1} + f_{ack}$. Fix such m'. Now we divide the path from $i_{e'}$ to i_d into intervals of length τ: let $d - e' = q\tau + r$, where q and r are nonnegative integers and $0 \leq r < \tau$. Assume that $q > 0$; the case where $q = 0$ is similar but simpler. Then by (1), we have $M'' \subseteq C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$, where $J = \{i_{e'+1}, \ldots i_{e'+\tau}\}$. If $m' \in C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$, then $get(m')_{i_{e'+\tau}}$ occurs by time $t_{e'+\tau+1,l-1} + f_{ack}$. Otherwise, we apply the $PC_{i_{e'}, i_{e'+\tau}}(\tau)$ condition, with $m = m'$ and $t = t_{e'+\tau+1,l-1} + f_{ack}$. This yields $m_1 \notin C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$ such that $get(m_1)_{i_{e'+\tau}}$ occurs by time $t_{e'+\tau+1,l-1} + f_{ack} + (\gamma_1\tau + \gamma_2\tau)f_{prog} \leq t_{e'+2\tau+1,l-1} + f_{ack}$. We show that $m_1 \in M' - M''$. Thus, in either case, there exists $m_1 \in M' - M''$ such that $get(m_1)_{i_{e'+\tau}}$ occurs by time $t_{e'+2\tau+1,l-1} + f_{ack}$. We repeat the same argument using $PC_{i_{e'+\tau}, i_{e'+2\tau}}(\tau)$, $PC_{i_{e'+2\tau}, i_{e'+3\tau}}(\tau), \ldots, PC_{i_{e'+(q-1)\tau}, i_{e'+q\tau}}(\tau)$, to show that there exists $m_q \in M' - M''$ such that $get(m_q)_{i_{d-r}}$ occurs by time $t_{d-r+\tau+1,l-1} + f_{ack}$. Then, by applying $PC_{i_{d-r},j}(\tau)$, we show that there exists $m'' \in M' - M''$ such that $get(m'')_j$ occurs by time $t_{d-r+\tau+1,l-1} + f_{ack} + (\gamma_1 r + \gamma_2\tau)f_{prog} \leq t_{d,l}$. This implies Statement 2 for j and l. □

8.2.2 Probabilistic Message Delivery Upper Bound

Now we prove a lower bound on the probability of the event $PC(\tau) \cap N$, and then tie all the results together in Theorem 8.12. In the next lemmas, our probabilistic statements are with respect to the system $BMMB \| Mac \| Env \| Net$, where *Mac* is an arbitrary implementation of the probabilistic layer with parameters f_{prog}, f_{ack}, ϵ_{prog}, and ϵ_{ack}, and *Env* is some probabilistic environment that submits at most k messages. The first lemma bounds the probability of fast message propagation between particular nodes i and j. Specifically, after any finite execution β in which i gets a new message, it lower-bounds the probability that either some new message is delivered to j within a short time, or else the execution is not *nice*. Here we use the conditional distribution on time-unbounded executions of *BMMB* that extend β. The notations A_β and $\Pr_\beta$ are defined in Section 2.

LEMMA 8.6. *Consider processes i and j, write $P_{i,j}$ as $i = i_0, i_1, i_2, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$. Let β be a finite execution of the BMMB protocol that ends at time t_0. Assume that there exists $m \notin C^{\beta}_{\Gamma(I)}(t_0)$ such that a $bcast(m)_i$ event occurs in β. Let F be the subset of A_β in which there exists $m' \notin C^{\beta}_{\Gamma(I)}(t_0)$ for which a $get(m')_j$ event occurs by time $t_0 + (\gamma_1 d + \gamma_2\tau)f_{prog}$. Then $\Pr_\beta(F \cup \bar{N}) \geq 1 - e^{-\tau}$.*

PROOF. See Lemmas 7.11 and 7.12 of [11]. We first prove a version of the lemma that includes a δ term to handle race conditions, and later remove δ. The proof follows the general outline of that for Lemma 7.5, and again uses Lemma 2.1. Now we use the path $P_{i,j}$, and define $t_q = t_0 + q(f_{prog} + \delta)$. The definitions of $Dist_q$ and X_q are similar to before, only now they talk about progress for *some message* not in $C^{\beta}_{\Gamma(I)}(t_0)$, rather than just the single given message. Arguments throughout the proof are modified to give progress guarantees for messages not in $C^{\beta}_{\Gamma(I)}(t_0)$. □

To prove a lower bound on the probability for PC, we define an alternative progress condition WPC ("weak progress condition"). WPC is defined in terms of intervals that begin with *get* and *ack* events, rather than intervals that begin at arbitrary points. We prove that WPC is equivalent to PC.

DEFINITION 8.7 ($WPC_{i,j,c}(\tau)$). *, Let $i, j \in V$, $i \neq j$ be two processes and let c be a positive integer. Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$. We say that $\alpha \in WPC_{i,j,c}(\tau)$ if for every nonnegative real t, the following holds: If α contains at least c get or ack events, and the c^{th} such event occurs at time t, and a $get(m)_i$ event for some message $m \notin C^{\alpha}_{\Gamma(I)}(t)$ occurs by time t, then a $get(m')_j$ event for some message $m' \notin C^{\alpha}_{\Gamma(I)}(t)$ occurs by time $t + (\gamma_1 d + \gamma_2\tau)f_{prog}$. We define the set $WPC(\tau)$ of executions as $\bigcap_{i,j,i\neq j,1\leq c\leq 2nk} WPC_{i,j,c}(\tau)$.*

LEMMA 8.8. $PC(\tau) = WPC(\tau)$.

We prove a lower bound for $WPC(\tau)$:

LEMMA 8.9. $\Pr(WPC(\tau) \cup \bar{N}) \geq 1 - 2n^3 k e^{-\tau}$.

PROOF. We use Lemma 8.6 to prove that, for every i, j, and c, $\Pr(WPC_{i,j,c}(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}$. A union bound yields the result. □

We then use Lemmas 8.9 and 8.8 to obtain a lower bound on the probability of $PC(\tau)$:

LEMMA 8.10. $\Pr(PC(\tau) \cap N) \geq 1 - 2n^3 k e^{-\tau} - \Pr(\bar{N})$.

We combine Lemma 8.5 with Lemma 8.10 and the bound for $\Pr(\bar{N})$ in Lemma 7.3, and instantiate τ as $\lceil \ln(\frac{2n^3 k}{\epsilon}) \rceil$, to obtain our result for *BMMB* over the probabilistic MAC layer:

THEOREM 8.11. *Let $m \in M$. Then BMMB guarantees that, with probability at least $1 - \epsilon - nk\epsilon_{ack}$, the following property holds of the generated execution α: Suppose an $arrive(m)_i$ event π occurs in α, and let t_0 be the time of occurrence of π. Let k' be a positive integer such that $|K(m)| \leq k'$. Then $deliver(m)$ events occur at all nodes in α by time*

$$t_0 + (k'-1)f_{ack} + \left((\gamma_1 + \gamma_2)D + \left((\gamma_1 + 2\gamma_2)\left\lceil \ln \frac{2n^3 k}{\epsilon} \right\rceil + \gamma_1 + \gamma_2\right) k\right) f_{prog}.$$

Finally, we combine the bound for *BMMB* in terms of the probabilistic MAC layer with the bound for *DMAC* to obtain a bound for *BMMB*-Decay. Our probabilistic statement is for the system $BMMB\text{-Decay}(\phi) \| Env \| Net$, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$, and *Env* is some probabilistic environment that submits at most k messages.

THEOREM 8.12. *Let $m \in M$. Let $\phi = \lceil 8\Delta \ln \frac{1}{\epsilon} \rceil$. Then BMMB-*Decay$(\phi)$ *guarantees that, with probability at least $1-\epsilon$, the following property holds for the generated execution α: Suppose an arrive$(m)_i$ event π occurs in α. Let k' be such that $|K(m)| \leq k'$. Then deliver(m) events occur at all nodes in α within time $O((D+k'\Delta \log \frac{nk}{\epsilon}) \log \Delta)$ of the time of occurrence of π.*

PROOF. Choose $\epsilon_{ack} = \frac{\epsilon}{2nk}$. Theorem 8.11 implies that, with probability at least $1 - \frac{\epsilon}{2} - nk\epsilon_{ack} \geq 1 - \epsilon$, $get(m)$ events occur everywhere within time

$$(k'-1)f_{ack} + \left((\gamma_1+\gamma_2)D + \left((\gamma_1+2\gamma_2)\left\lceil \ln \frac{4n^3k}{\epsilon} \right\rceil + \gamma_1 + \gamma_2\right)k'\right) f_{prog}.$$

Using the definitions of parameters for the implementation of the probabilistic layer, in Section 5, we may assume that $\epsilon_{prog} \leq \frac{7}{8}$, so this expression is $O((D + \log(\frac{nk}{\epsilon})k')f_{prog}) + (k'-1)f_{ack}$. Again using those parameter definitions, we substitute $f_{prog} = O(\log \Delta)$ and $f_{ack} = O(\Delta \log(\frac{nk}{\epsilon}) \log \Delta)$ into the expression, to get a bound of

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

To see why we can use $f_{ack} = O(\Delta \log(\frac{nk}{\epsilon}) \log \Delta)$ here, instantiate ϵ in the parameter definitions with $\frac{\epsilon}{2nk\Delta}$, for the ϵ in the statement of this theorem. Then the parameter definitions say that $\epsilon_{ack} = \frac{\epsilon}{2nk\Delta}\Delta = \frac{\epsilon}{2nk}$. This yields that $f_{ack} = O(\Delta \log(\frac{2nk\Delta}{\epsilon}) \log \Delta) = O(\Delta \log(\frac{nk}{\epsilon}) \log \Delta)$, as needed. □

9. CONCLUSIONS

We have shown how one can use abstract MAC layers to decompose broadcast algorithms into a high-level part for broadcast and a low-level part for contention management. We use both the basic abstract MAC layer of [14, 15] and a new probabilistic layer. The basic layer is simple to use, but yields bounds that are not quite optimal. The probabilistic layer yields better bounds, at the cost of somewhat harder high-level analysis. The approach is flexible, in that it allows high-level algorithms to be combined easily with different implementations of the MAC layer.

Our analysis of the high-level multi-message broadcast algorithm is sufficiently hard that we think it would have been infeasible without such a decomposition. Thus, we believe that this approach enables analysis of more complicated algorithms than one could handle otherwise. Current and future work involves designing and analyzing other algorithms over the MAC layers, and developing other algorithms to implement the MAC layers. We hope that this work will contribute to building a comprehensive theory for wireless network algorithms, spanning from the physical level to applications.

10. REFERENCES

[1] M. Adler and C. Scheideler. Efficient communication strategies for ad hoc wireless networks. *Theory Comput. Syst.*, 33(5/6):337–391, 2000.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[3] R. Bar-Yehuda, A. Israeli, and A. Itai. Multiple communication in multi-hop radio networks. *SIAM Journal on Computing*, 22(4):875–887, 1993.

[4] A. Cornejo, N. Lynch, S. Viqar, and J. Welch. A neighbor discovery service using an abstract MAC layer. In *Proceedings of Allerton Conference on Communication, Control and Computing*, 2009.

[5] A. Cornejo, S. Viqar, and J. Welch. Reliable Neighbor Discovery for Mobile Ad Hoc Networks In *Proceedings of the International Workshop on Foundations of Mobile Computing*, 2010. To appear.

[6] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 492–501, 2003.

[7] S. Dolev, S. Gilbert, M. Khabbazian, and C. Newport. Broadcasting in radio networks with multiple channels. Submitted for publication.

[8] L. Gasieniec. On efficient gossiping in radio networks. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity*, pages 2–14, 2009.

[9] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *Proceedings of the ACM SIGCOMM Conference*, volume 38, pages 159–170, 2008.

[10] T. Jurdzinski and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks. In *Proceedings of International Symposium on Algorithms and Computation*, pages 139–150, 2002.

[11] M. Khabbazian, D. Kowalski, F. Kuhn, and N. Lynch. The cost of global broadcast using abstract MAC layers. *MIT Tech. Report (MIT-CSAIL-TR-2010-005)*, February 2010.

[12] M. Khabbazian, N. Lynch, M. Medard, and A. Parandeh-Gheibi. MAC design for analog network coding. *MIT Tech. Report (MIT-CSAIL-TR-2010--036)*, July 2010.

[13] D. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 73–82, 2003.

[14] F. Kuhn, N. Lynch, and C. Newport. The abstract MAC layer. In *Proceedings of the International Symposium on Distributed Computing*, pages 48–62, 2009.

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

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

[17] A. Pelc. Algorithmic aspects of radio communication. In *Proceedings of the International Workshop on Foundations of Mobile Computing*, pages 1–2, 2008.

[18] D. Peleg. Time-efficient broadcasting in radio networks: A review. In *Proceedings of the International Conference on Distributed Computing and Internet Technologies*, pages 1–18, 2007.

[19] J. Walter, J. Welch, and N. Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. *Wireless Networks*, 7(6):585–600, 2001.