Decomposing Broadcast Algorithms Using Abstract MAC Layers

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

Dariusz Kowalski[2]
University of Liverpool
United Kingdom
d.kowalski@liverpool.ac.uk

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

Nancy Lynch[1]
Massachusetts Institute of Technology
United States
lynch@csail.mit.edu

Abstract

In much of the theoretical literature on global broadcast algorithms for wireless networks, 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 more difficult communication problems. In this paper, we present results aimed at simplifying such algorithms and analysis by decomposing the treatment into two levels, using abstract "MAC layer" specifications to encapsulate contention management. We use two different abstract MAC layers: the basic layer of [1, 2] and a new probabilistic layer.

We first present a typical randomized contention-management algorithm for a standard graph-based radio network model and show that it implements both abstract MAC layers. Then we combine this algorithm with greedy algorithms for single-message and multi-message global broadcast and analyze the combinations, 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.

Keywords: Broadcast protocol, global broadcast, 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, for example, by [3], analyzes wireless network 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 [4], uses models that are close to the actual physical network and requires all algorithms to handle basic communication issues.

[1]Research supported by AFOSR contract FA9550-08-1-0159 and NSF grants CCF-0726514, CNS-0715397, CCF-0937274, and NSF-PURDUE-STC Award 0939370-CCF.

[2]Research supported by the Engineering and Physical Sciences Research Council [grant numbers EP/G023018/1, EP/H018816/1].

Ignoring MAC layer issues and working with high-level communication models makes it possible to design and analyze complex algorithms for high-level problems. The analysis of typical information-dissemination protocols for tasks like single-message and multi-message message broadcast become almost trivial. However, such analysis may not be entirely 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, on the other hand, avoids these problems, but requires the algorithm designer to cope with physical layer issues such as message loss due to interference and 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 (e.g., [5, 6, 7]), requiring algorithms to be rewritten and reanalyzed for each new model. This complexity is an impediment to the development of a theory for wireless network algorithms.

Recently, Kuhn et al. proposed a new approach with the goal of combining the advantages of both previous approaches, while avoiding their major problems [1, 2, 8, 9]. 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 abstract MAC layer over a physical network, and one that uses the abstract 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. This work demonstrated how one might build a theory for high-level wireless network algorithms that does not ignore issues of contention, but makes analysis of high-level algorithms tractable.

Kuhn et al. focused on high-level issues of designing and analyzing algorithms over the abstract 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 MAC-layer 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.

This paper. 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 greedy protocols for broadcasting a single message and multiple messages throughout a wireless 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 MAC layers: the basic (non-probabilistic) one from [1, 2], and a new probabilistic layer.

For contention management, we use a randomized algorithm called *DMAC* that is similar to those in [4, 10]; in this algorithm, nodes transmit repeatedly using a predetermined schedule of transmission probabilities. We show that *DMAC* implements the basic abstract MAC layer with high probability, and that it implements the probabilistic 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 [4]. Our combined algorithm is similar to that of [4]; the key difference is that we decompose the algorithm and its 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 single 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 $O((D + k'\Delta\log(\frac{n}{\epsilon}))\log(\Delta))$, respectively. Our analysis for multi-message broadcast over the probabilistic layer is not easy; in fact, we believe it would be infeasible without such a decomposition.
Note that, for both our single-message and multi-message broadcast algorithms, the bounds that we have obtained using the new probabilistic MAC layer are better than those using the basic MAC layer. When we began this work, we first considered just the basic layer, as in [1, 2], and obtained our bounds for broadcast as easy corollaries of results already proved in [1, 2]. However, the bound we obtained for single-message broadcast was not quite as good as the best known bound (in [4]), which led us to define the probabilistic layer and reanalyze the high-level broadcast algorithms using that layer.

Discussion. The main contributions of this paper are: (1) the definition of the new probabilistic MAC layer, (2) the clean decomposition of a single-message broadcast algorithm similar to that of Bar-Yehuda et al. [4] into two pieces, a greedy high-level protocol and the *DMAC* contention-management algorithm, which can be used and analyzed independently, and (3) the design and analysis of a multi-message broadcast algorithm based on the broadcast algorithm of [1, 2] 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. [11, 12] have developed new Neighbor Discovery algorithms over the basic abstract MAC layer. These enable the construction of high-level dynamic graph models like the one used in [3] over an abstract MAC layer, which supports the analysis of many dynamic graph algorithms in terms of abstract MAC layers, and therefore, in terms of physical network models. Also, Dolev et al. [13] 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. Also, Khabbazian et al. [14] have developed an implementation of the probabilistic abstract MAC layer based on Analog Network Coding (ANC) techniques [15]. This implementation yields better bounds than the implementation in this paper, under certain assumptions. By combining their implementation with our high-level multi-message broadcast algorithm, they obtain an algorithm and a complexity bound for multi-message broadcast using ANC.

Related work. This work relies on [1, 2] for the general idea of decomposing wireless network algorithms using an abstract MAC layer, as well as the basic abstract MAC layer specification and the greedy multi-message global broadcast algorithm. Later versions of this work appear as [8, 9]. The later versions include some small improvements, including removing a technical assumption that all messages sent on the MAC layer are unique. Adler and Scheideler [16] 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 [4]. That paper contains a randomized algorithm that accomplishes the task in $O((D+\log(\frac{n}{\epsilon}))\log(\Delta))$ 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., [17, 18] obtain a bound of $O((D+\log(\frac{n}{\epsilon}))\log(\frac{n}{D}))$, which improves upon [4] 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 [19]; 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 $O((n + (k + D)\log(\frac{n}{\epsilon}))\log(\Delta))$, 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 [18] contains a randomized algorithm for n simultaneous broadcasts working in time $O(n\log(\frac{n}{\epsilon})\log(n))$ with probability $\geq 1-\epsilon$. This algorithm differs from ours and that of [19] in that it allows intermediate nodes to combine an arbitrary amount of information into a single message, thus reducing high-level contention.

In all of this prior work on broadcast, the issues involving broadcast and contention management are intermingled. Earlier versions of this work appeared in [20, 21].

The rest of the paper is organized as follows. Section 2 describes mathematical preliminaries. Section 3 presents our physical network assumptions. Section 4 presents our two abstract MAC layers. Section 5 presents a probabilistic algorithm that implements both of our abstract MAC layers over the physical network. Section 6 defines the global broadcast problem and our broadcast algorithms. Section 7 presents our results for single-message broadcast, and Section 8 our results for multi-message broadcast. Section 9 concludes.

2. Mathematical Preliminaries

We collect here some necessary mathematical background related to graph theory, probability distributions, and probabilistic timed I/O automata.

2.1. Graph Theory

Throughout this paper, we fix a (static) connected undirected network 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 $\sigma = \lceil \log(\Delta + 1) \rceil$ slots. 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 $\Gamma(I) = \bigcup_{i \in I} \Gamma(i)$.

Definition 2.1 (Consistent shortest paths)**.** *For every $i, j \in V$, we fix a shortest path $P_{i,j}$ from i to j in G. We assume that these shortest paths are* consistent *in the sense that, 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 $P_{i,j}$.*

One way to obtain a consistent set of shortest paths is to define a total order on the nodes in V, regard a path as a sequence of nodes, and define $P_{i,j}$ to be the lexicographically smallest shortest path from i to j.

2.2. Probability Distributions

The following simple lemma compares probability distributions. It is used twice later, in the proofs of Lemmas 7.3 and 8.11.

Lemma 2.2. *For every positive integer q, let X_q and Y_q be $\{0, 1\}$-valued random variables. Suppose that the Y_q are a collection of independent random variables. Suppose further that:*

1. *$Pr(X_1 = 1) \geq Pr(Y_1 = 1)$.*
2. *For every $q \geq 2$ and $x_1, x_2, ..., x_{q-1} \in \{0, 1\}$, $Pr(X_q = 1 | X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \geq Pr(Y_q = 1)$.*

Then for every $r \geq 1$ and every nonnegative integer d,

$$Pr(\sum_{q=1}^{r} X_q \geq d) \geq Pr(\sum_{q=1}^{r} Y_q \geq d).$$

Proof. By induction on r. The base case, for $r = 1$, follows from the first enumerated assumption. For the inductive step, suppose the result holds for $r \geq 1$ and show it for $r + 1$. We have that

$$\begin{aligned}
Pr(\sum_{q=1}^{r+1} X_q \geq d) &= Pr(X_{r+1} = 1 | \sum_{q=1}^{r} X_q = d - 1) \cdot Pr(\sum_{q=1}^{r} X_q = d - 1) + Pr(\sum_{q=1}^{r} X_q \geq d) \\
&\geq Pr(Y_{r+1} = 1) \cdot Pr(\sum_{q=1}^{r} X_q = d - 1) + Pr(\sum_{q=1}^{r} X_q \geq d) \\
&= Pr(Y_{r+1} = 1) \cdot (Pr(\sum_{q=1}^{r} X_q \geq d - 1) - Pr(\sum_{q=1}^{r} X_q \geq d)) + Pr(\sum_{q=1}^{r} X_q \geq d) \\
&= Pr(Y_{r+1} = 1) \cdot Pr(\sum_{q=1}^{r} X_q \geq d - 1) + Pr(Y_{r+1} = 0) \cdot Pr(\sum_{q=1}^{r} X_q \geq d).
\end{aligned}$$

By the inductive hypothesis on r, for both $d-1$ and d, we get that this last expression is greater than or equal to

$$\begin{aligned}
&Pr(Y_{r+1}=1)\cdot Pr(\sum_{q=1}^{r} Y_q \geq d-1) + Pr(Y_{r+1}=0)\cdot Pr(\sum_{q=1}^{r} Y_q \geq d)\\
&= Pr(Y_{r+1}=1)\cdot Pr(\sum_{q=1}^{r} Y_q = d-1) + Pr(\sum_{q=1}^{r} Y_q \geq d)\\
&= Pr(\sum_{q=1}^{r+1} Y_q \geq d).
\end{aligned}$$

Combining all the inequalities, we get

$$Pr(\sum_{q=1}^{r+1} X_q \geq d) \geq Pr(\sum_{q=1}^{r+1} Y_q \geq d),$$

as needed to complete the inductive step. □

The following lemma encapsulates a Chernoff bound analysis. It is used in the proofs of Lemmas 7.3 and 8.11.

Lemma 2.3. *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 $r = \lfloor \frac{1}{p}(3d + 2\tau) \rfloor$. Then*

$$Pr(\sum_{q=1}^{r} Y_q < d) \leq e^{-\tau} \ .$$

Proof. Let $\mu = rp$. Using Chernoff, we get:

$$Pr(\sum_{q=1}^{r} Y_q < d) \leq \exp\left(-\frac{1}{2}\frac{(\mu-d)^2}{\mu}\right). \tag{1}$$

Note that the function $f(x) = \exp(-\frac{(x-d)^2}{2x})$ is non-increasing in x for $d \leq x$. Also, since $d \geq 1$, we have

$$d \leq 3d + 2\tau - p = \left(\frac{1}{p}(3d+2\tau) - 1\right) p \leq \left\lfloor \frac{1}{p}(3d+2\tau) \right\rfloor p = rp = \mu \ .$$

Therefore,

$$\begin{aligned}
\exp\left(-\frac{1}{2}\frac{(\mu-d)^2}{\mu}\right) &\leq \exp\left(-\frac{1}{2}\frac{(3d+2\tau-p-d)^2}{3d+2\tau-p}\right)\\
&= \exp\left(-\frac{1}{2}\frac{(2d+2\tau-p)^2}{3d+2\tau-p}\right)\\
&\leq \exp\left(-\tau\right) \ .
\end{aligned}$$

□

2.3. Probabilistic Timed I/O Automata (PTIOA)

We formalize our results in terms of *probabilistic timed I/O automata*, as defined by Mitra [22]. PTIOAs include mechanisms (local schedulers and task schedulers) to resolve nondeterminism.[1]

[1]Here, we modify Mitra's model slightly: We assume that the task scheduler is a mapping that takes each finite set of tasks of size ≥ 2 to an infinite sequence of individual tasks in the set. This task scheduler is used to resolve nondeterministic choices whenever a set of two or more tasks (which are sets of locally-controlled actions) contain actions that are enabled at the same time.

Throughout the paper, 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. In addition, we often consider probabilities with respect to a "cone" in the full probabilistic execution following a particular closed execution β.[2] 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_β.

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 t_{slot}; for simplicity, we assume that $t_{slot} = 1$. Processes have synchronized clocks, and so can detect when each slot begins and ends. Processes communicate only on slot boundaries. We assume all processes awaken at the same time 0, which is the beginning of slot 1. We assume that each node 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 $G = (V, E)$, one per node. Following a common conventions, we sometimes ignore the distinction between processes and the graph nodes at which they reside, referring to processes as "nodes". Our model is a special case of the model considered in [1, 2], with only a single, static, undirected graph G. Processes know n and Δ, but nothing else about the graph; in particular, they do not know their neighbors in G.

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 its G-neighboring processes. Thus, each process j, in each slot, is reached by some collection of messages (from itself and its transmitting neighbors). What process j actually *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. Thus, a process always receives its own message, regardless of what else reaches it. (a) If j is not reached by its own message, but is reached by exactly one message (from another process), then it receives that message. (b) If j is reached by no messages, it receives silence, represented by $\perp$. (c) If j is not reached by its own message, but is reached by two or more messages from other processes, then it receives silence, $\perp$. Thus, processes cannot distinguish collisions from silence; that is, we assume no collision-detection.

4. Abstract MAC Layers

In this section, we specify the two abstract MAC layers, a special case of the basic layer of [1, 2],[3] and the new probabilistic layer. Our layers are defined for a single, static, undirected communication graph $G = (V, E)$ with maximum node degree Δ; this is a special case of the layer in [1, 2], which allows two graphs, G and G', representing guaranteed and possible communication.

Both of our 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 $i \in V$. Both specifications are parameterized by positive reals, f_{rcv}, f_{ack}, and f_{prog}. These bound delays for a particular message to arrive at a particular receiver, for an acknowledgement to arrive at a sender indicating that its message has arrived at all neighbors, and for some message from among many competing messages to arrive at a receiver.[4] For many MAC implementations, f_{prog} is notably smaller than f_{rcv} and f_{ack}, because the time for *some* message to arrive at a receiver is substantially shorter than the time for a particular message to arrive at *every* neighbor. Both specifications also use a (small) nonnegative real parameter t_{abort}, which

[2] "Closed" means that the execution is a finite sequence of alternating discrete and continuous steps, and the final continuous step spans a closed time interval.

[3] The definition of the basic layer is slightly more general in later versions of this work [8, 9], in that it drops a "unique message" assumption. Here we retain that assumption, since it makes some things a bit simpler.

[4] Since our bounds do not depend on the actual contention, but only on maximum node degree, we express them as constants rather than as functions of the contention as in [1, 2].

bounds the amount of time after a sender aborts a sending attempt when the message could still arrive at some receiver.

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 $Mac \| Env \| Net$ (again 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 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 $bcast(m)_i$. The *cause* function 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 $abort(m)_i$.

For any α that is is well-formed and satisfies the above restrictions, we define a *message instance* in α to be 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 particular 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. The receive delay bound also includes another constraint, bounding the time after an *abort* when a corresponding *rcv* may occur.

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 receive delay, acknowledgement delay, and progress. In addition to the four parameters above (f_{rcv}, f_{ack}, f_{prog}, and t_{abort}), this specification uses parameters ϵ_{rcv}, ϵ_{ack}, and ϵ_{prog}, representing error probabilities for satisfying the delay bounds.

The Probabilistic Abstract MAC Layer 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 the *Guaranteed communication* and *Termination* properties do not appear in this list; we replace these with probabilistic versions, in the acknowledgement delay bound, below.
The specification also says that the *Mac* automaton must guarantee the following three probabilistic upper bounds on message delays. In defining these bounds, we use the following terminology: 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.
   - 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.
   - 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 $i \in I$.
   - B, the executions in which, by time $t + f_{prog}$, at least one of the following occurs:
     (a) An $ack(m_i)_i$ for every $i \in I$,
     (b) A $rcv(m_i)_j$ for some $i \in I$, or
     (c) 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 process 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.

4.3. Nice Executions

We end Section 4 with a technical definition and lemma related to our acknowledgement delay bound. These are used in our analysis in both Sections 7 and 8. The "executions" considered here are executions of any system of the form $Mac \| Env \| Net$, where *Mac* implements the probabilistic abstract MAC layer with acknowledgement parameters f_{ack} and ϵ_{ack}, *Env* is a well-formed probabilistic environment for the MAC layer and *Net* is the physical layer.

Definition 4.1 (*Nice* broadcast events and *nice* executions)**.** *Suppose a $bcast(m)_i$ event π occurs at time t_0 in execution α. Then we say that 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. We say that execution α is* nice *if all bcast events in α are nice. Let N be the set of all nice executions.*

The following lemma bounds the probability that an execution is not nice.

Lemma 4.2. *If Env submits at most b bcasts in any execution and never submits an abort, then*

$$Pr(\bar{N}) \leq b \cdot \epsilon_{ack}.$$

Proof. For any integer b', define:

- $H_{b'}$ to be the set of time-unbounded executions that contain at least b' *bcast* events, and in which it is not the case that, by time f_{ack} after the $(b')^{th}$ *bcast* event, a corresponding *ack* occurs that is preceded by a corresponding *rcv* for every neighbor of the broadcasting node.
- $C_{b'}$ to be the set of time-unbounded executions that contain strictly fewer than b' *bcast* events.
- $B_{b'}$ to be the set of finite executions β such that β is a prefix of a time-unbounded execution that contains at least b' *bcast* events and β ends with the $(b')^{th}$ *bcast* event.

Then $\bar{N} = \bigcup_{b',1 \leq b' \leq b} H_{b'}$; the bound b suffices because each execution contains at most b *bcast* events. Also, $C_{b'} \subseteq \bar{H}_{b'}$. Also, the sets $\{A_\beta\}_{\beta \in B_{b'}}$ and $C_{b'}$ constitute a partition of the set of all time-unbounded executions. (The notation A_β is defined in Section 2.3.)
For each $\beta \in B_{b'}$, the definition of f_{ack} implies that

$$Pr_\beta(H_{b'}) \leq \epsilon_{ack}.$$

Then we obtain:

$$\begin{aligned} Pr(H_{b'}) &= \Sigma_{\beta \in B_{b'}}(Pr_\beta(H_{b'}) \cdot Pr(A_\beta)) + Pr(H_{b'}|C_{b'}) \cdot Pr(C_{b'}) \\ &= \Sigma_{\beta \in B_{b'}}(Pr_\beta(H_{b'}) \cdot Pr(A_\beta)) \\ &\leq \Sigma_{\beta \in B_{b'}}(\epsilon_{ack} \cdot Pr(A_\beta)) \\ &\leq \epsilon_{ack}. \end{aligned}$$

Then, using a union bound, we obtain:

$$Pr(\bar{N}) = Pr(\bigcup_{b',1 \leq b' \leq b} H_{b'}) \leq b \cdot \epsilon_{ack},$$

as needed. □

5. Implementing the Abstract MAC Layer over the Physical Layer

We implement our abstract MAC layers using a contention management algorithm *DMAC*. *DMAC* uses a probabilistic transmission strategy similar to the *Decay* strategy of [4] and the *Probability-Increase* strategy of [10]. We prove two theorems about *DMAC*: Theorem 5.7 says that *DMAC* implements the probabilistic abstract MAC layer (exactly), and Theorem 5.13 says that it implements the basic abstract MAC layer with high probability.

5.1. Modified Decay Algorithm

Our Decay probabilistic transmission algorithm differs slightly from the one in [4] in that the processes successively *increase* their transmission probabilities in each Decay phase rather than decrease them.[5] Also, in our algorithm the processes choose randomly whether to transmit in each individual slot, whereas in [4], they choose randomly whether to drop out of the entire 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.

[5]Thus, our strategy might be better called "Growth" rather than "Decay", but we keep the original name.

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 our modified* 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. This depends on the following claim:

Claim 1: At some slot s, the number of participants c_s satisfies $\frac{1}{2c_s} \leq p_s \leq \frac{1}{c_s}$.

Proof of Claim 1: We must have $p_1 \leq \frac{1}{c_1}$, because if not, then $p_1 > \frac{1}{c_1}$, which means that $\frac{1}{2^{\lceil \log(\Delta+1) \rceil}} > \frac{1}{c_1} \geq \frac{1}{\Delta+1}$. This implies that $\frac{1}{\Delta+1} > \frac{1}{\Delta+1}$, a contradiction. If also $\frac{1}{2c_1} \leq p_1$, then we are done. So assume that $p_1 < \frac{1}{2c_1}$. Then it must be that $p_2 < \frac{1}{c_2}$, because $p_2 = 2p_1$ and $c_2 \leq c_1$. Again, if also $\frac{1}{2c_2} \leq p_2$, we are done, so assume that $p_2 < \frac{1}{2c_2}$. We continue in this way through all the slots. If we never reach one where we are done, it must be that $p_\sigma < \frac{1}{2c_\sigma}$. However, $p_\sigma = \frac{1}{2}$ and $c_\sigma \geq 1$, so this is impossible.

Given Claim 1, we consider what happens at the indicated slot s. If j participates in slot s, then the probability that some process in I transmits alone is exactly $(c_s - 1)p_s(1 - p_s)^{c_s - 1}$. This is at least $(c_s - 1)(\frac{1}{2c_s})(1 - \frac{1}{c_s})^{c_s - 1} = \frac{1}{2}(\frac{c_s - 1}{c_s})(1 - \frac{1}{c_s})^{c_s - 1} = \frac{1}{2}(1 - \frac{1}{c_s})^{c_s}$ We have that $c_s \geq 2$, because at least one process in I participates in slot s, in addition to j. So $\frac{1}{2}(1 - \frac{1}{c_s})^{c_s} \geq \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$. Thus, if j participates in slot s, the probability that some process in I transmits alone is at least $\frac{1}{8}$.
On the other hand, if j does not participate in slot s, then the probability that some process in I transmits alone is exactly $c_s p_s (1 - p_s)^{c_s - 1}$. If $c_s = 1$, then this is equal to p_s, which is $\geq \frac{1}{2c_s} = \frac{1}{2}$. If $c_s > 1$, then the value of the expression is at least $c_s(\frac{1}{2c_s})(1 - \frac{1}{c_s})^{c_s - 1} = \frac{1}{2}(1 - \frac{1}{c_s})^{c_s - 1}$, which is $\geq \frac{1}{2}(1 - \frac{1}{c_s})^{c_s} \geq \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$. Thus, if j does not participate in slot s, then the probability that some process in I transmits alone is at least $\frac{1}{8}$. □

5.2. The DMAC Algorithm

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

DMAC(ϕ), ϕ a positive integer:
We group slots into Decay phases, each consisting of exactly σ slots.

Each MAC layer process i that receives a message from its environment, via a $bcast(m)_i$ input 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 $abort(m)_i$ input from the environment before it performs $ack(m)_i$, then 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 message to its environment with a $rcv(m')_i$ output event, at a real time slightly 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.

5.3. Properties of DMAC

Throughout this subsection, we let *Env* be any probabilistic environment, and consider the unique probabilistic execution of $DMAC(\phi) \| Env \| Net$. We prove five lemmas giving properties of $DMAC(\phi)$.
First, Lemma 5.2 asserts that $DMAC(\phi)$ satisfies all of the non-probabilistic guarantees.

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. Moreover, no rcv happens more than time* 1 *after a corresponding abort.*

Proof. Straightforward. For the last property, note that when a message is aborted, its transmitting process i participates in no further slots for that message. That implies that the lag time is at most $t_{slot} = 1$. □

The next lemma gives an absolute bound on acknowledgement time.

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 $bcast(m)_i$.*

Proof. Immediate from the definition of $DMAC(\phi)$. □

The rest of the section gives probabilistic properties. First, we apply Lemma 5.1 to obtain a probabilistic version of the progress bound.

Lemma 5.4. *Let $j \in V$. Let β be a closed execution that ends at time t. Let I be the set of neighbors of j that have active bcasts at the end of β, where $bcast(m_i)_i$ is the bcast at i. Suppose that I is nonempty. Suppose that no $rcv(m_i)_j$ occurs in β, for any $i \in I$. Let g and h be nonnegative integers, $h > 0$. Let $g + 1$ be the number of the first* Decay *phase that starts strictly after time t. Define the following sets of time-unbounded executions that extend β:*

- *A, the executions in which no $abort(m_i)_i$ occurs for any $i \in I$.*
- *B, the executions in which, by the end of* Decay *phase $g + h$, 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.*
- *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

1. $Pr_\beta(B \cup C|A) \geq 1 - (\frac{7}{8})^h$.
2. $Pr_\beta(\bar{A} \cup B \cup C) \geq 1 - (\frac{7}{8})^h$.

Proof. We have that

$$\begin{aligned} Pr_\beta(B \cup C|A) &= Pr_\beta(B \cup C|C \cap A)Pr_\beta(C|A) + Pr_\beta(B \cup C|\bar{C} \cap A)Pr_\beta(\bar{C}|A) \\ &= Pr_\beta(C|A) + Pr_\beta(B|\bar{C} \cap A)Pr_\beta(\bar{C}|A) \\ &\geq Pr_\beta(B|\bar{C} \cap A)(Pr_\beta(C|A) + Pr_\beta(\bar{C}|A)) \\ &= Pr_\beta(B|\bar{C} \cap A), \end{aligned}$$

so, for the first conclusion, it suffices to show that $Pr_\beta(B|\bar{C} \cap A) \geq 1 - (\frac{7}{8})^h$.
So assume $\bar{C} \cap A$, that is, that within h phases, not every $i \in I$ has an $ack(m_i)_i$, and no $abort(m_i)_i$ occurs for any $i \in I$. Then some neighbor of j participates in all phases q, where $g + 1 \leq q \leq g + h$. Then Lemma 5.1 implies that, in each phase q, (regardless of what has happened in the previous phases), the following holds: With probability $\geq \frac{1}{8}$, a rcv_j for a message m_i, $i \in I$, or for a "new" message (one whose *bcast* occurs after β), occurs at phase q, unless such a rcv_j occurs at an earlier phase. Thus,

$$Pr_\beta(B|\bar{C} \cap A) \geq 1 - \left(\frac{7}{8}\right)^h,$$

as needed.
The second conclusion follows from the first since

$$\begin{aligned} Pr_\beta(\bar{A} \cup B \cup C) &= Pr_\beta(\bar{A} \cup B \cup C|A)Pr_\beta(A) + Pr_\beta(\bar{A} \cup B \cup C|\bar{A})Pr_\beta(\bar{A}) \\ &= Pr_\beta(B \cup C|A)Pr_\beta(A) + Pr_\beta(\bar{A}) \\ &\geq Pr_\beta(B \cup C|A). \end{aligned}$$

□

The next lemma gives a probabilistic bound on the receive delay.

Lemma 5.5. *Let ϵ be a positive real. Suppose that ϕ, the parameter for DMAC(ϕ), is equal to $\lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$. 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. Let $g+1$ be the number of the first* Decay *phase that starts strictly after time t. Define the following sets of time-unbounded executions that extend β:*

- *A, the executions in which no $abort(m)_i$ occurs.*
- *B, the executions in which, by the end of* Decay *phase $g + \phi$, a $rcv(m)_j$ occurs.*

If $Pr_\beta(A) > 0$, then

1. *$Pr_\beta(B|A) \geq 1 - \epsilon$.*
2. *$Pr_\beta(\bar{A} \cup B) \geq 1 - \epsilon$.*

Proof. For every $q = 1, \ldots, \phi$, we define 0-1 valued random variable X_q by

$$X_q = \begin{cases} 1 & \text{if } rcv(m)_j \text{ occurs by the end of phase } g + q, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For the first conclusion, it suffices to show that

$$Pr_\beta(\sum_{q=1}^{\phi} X_q \geq 1|A) \geq 1 - \epsilon,$$

that is, that

$$Pr_\beta(\sum_{q=1}^{\phi} X_q = 0|A) \leq \epsilon.$$

First, we claim that

$$Pr_\beta(X_1 = 1|A) \geq \frac{1}{8\Delta}. \tag{3}$$

This is because, by Lemma 5.1, the conditional probability that some rcv_j occurs in phase $g + 1$ is at least $\frac{1}{8}$, and because all neighboring senders are equally likely to succeed. Similarly, for every q, $2 \leq q \leq \phi$, and $x_1, x_2, ..., x_{q-1} \in \{0, 1\}$,

$$Pr_\beta(X_q = 1|X_1 = x_1, \ldots, X_{q-1} = x_{q-1}, A) \geq \frac{1}{8\Delta}. \tag{4}$$

Then

$$
\begin{aligned}
&Pr_\beta(\sum_{q=1}^{\phi} X_h = 0|A) \\
&\qquad = Pr_\beta(X_1 = 0|A) \cdot Pr_\beta(X_2 = 0|X_1 = 0, A) \cdot Pr_\beta(X_3 = 0|X_1 = X_2 = 0, A) \\
&\qquad \cdot \ldots \cdot Pr_\beta(X_\phi = 0|X_1 = X_2 = \ldots = X_\phi = 0, A) \\
&\qquad \leq \left(1 - \frac{1}{8\Delta}\right)^{\phi} \\
&\qquad = \left(1 - \frac{1}{8\Delta}\right)^{\lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil} \\
&\qquad \leq \left(1 - \frac{1}{8\Delta}\right)^{8\Delta \ln(\frac{1}{\epsilon})} \\
&\qquad \leq e^{-\ln(1/\epsilon)} = \epsilon.
\end{aligned}
$$

The last inequality follows from $(1 + x) < e^x$. The second conclusion follows from the first as in the proof of Lemma 5.4, this time by showing that $Pr_\beta(\bar{A} \cup B) \geq Pr_\beta(B|A)$. □

The fifth and final lemma gives a probabilistic bound on the acknowledgement delay.

Lemma 5.6. *Let ϵ be a positive real. Suppose that $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil$. Let $i \in V$. Let β be any closed execution that ends with $bcast(m)_i$ at time t. Let $g + 1$ be the number of the first* Decay *phase that starts strictly after time t. Define the following sets of time-unbounded executions that extend β:*

- *A, the executions in which no $abort(m)_i$ occurs.*
- *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

1. *$Pr_\beta(B|A) \geq 1 - \epsilon\Delta$.*
2. *$Pr_\beta(\bar{A} \cup B) \geq 1 - \epsilon\Delta$.*

Proof. For the first conclusion, note that Lemma 5.3 implies that $ack(m)_i$ is certain to occur by the claimed time, in fact, just at the end of phase $g + \phi$. For the $rcv(m)_j$ events, we use Lemma 5.5 to conclude that the probability that each individual $rcv(m)_j$ event occurs within ϕ phases is $\geq 1 - \epsilon$. Then we use a union bound to conclude that the probability that all the $rcv(m)_j$ events occur within ϕ phases is $\geq 1 - \epsilon\Delta$.
The second conclusion follows as in the two previous proofs. □

5.4. Implementing the Probabilistic Layer

In this section, we show that $DMAC(\phi)$, for a particular choice of ϕ, implements the probabilistic abstract MAC layer. This implementation claim is precise—no new probabilities are introduced for the implementation relationship.
For this section, we fix several constants:

- ϵ, a real number, $0 < \epsilon \leq 1$.
- h, a positive integer. This is the number of Decay phases we will consider for the progress bound.
- $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil$. This is the number of Decay phases we will consider for the receive and acknowledgement bounds.

We define the seven parameters for the probabilistic MAC layer, as functions of ϵ, h, and ϕ:

- $f_{rcv} = f_{ack} = (\phi + 1)\sigma$.

- $f_{prog} = (h+1)\sigma$.
- $\epsilon_{rcv} = \epsilon$.
- $\epsilon_{ack} = \epsilon\Delta$.
- $\epsilon_{prog} = (\frac{7}{8})^h$.
- $t_{abort} = 1$.

Using Lemmas 5.2- 5.6, we obtain:

Theorem 5.7. *$DMAC(\phi)$ implements the Probabilistic Abstract MAC Layer with parameters as defined above.*

Proof. We consider a system consisting of $DMAC(\phi)$ composed with a well-formed probabilistic environment *Env* and the physical network *Net*. We assume that (after all nondeterminism in *Net* and in the scheduling is suitably resolved) that the composition $DMAC(\phi)\|Env\|Net$ yields a single probabilistic execution. We must show that this probabilistic execution satisfies all of the non-probabilistic and probabilistic guarantees that are listed in Section 4.2.
Lemma 5.2 implies immediately that the probabilistic execution satisfies all of the needed non-probabilistic guarantees.
Lemma 5.5, Conclusion 1, implies that the probabilistic execution satisfies the first probabilistic requirement, on the receive delay. In some detail, consider any closed execution that ends with a $bcast(m)_i$ at time t, and define A and B as in the definition of the receive delay bound, where $f_{rcv} = (\phi+1)\sigma$ and $\epsilon_{rcv} = \epsilon$. Suppose that $Pr_\beta(A) > 0$. Let $g+1$ be the number of the first Decay phase that starts strictly after time t. Define B' to be the set of time-unbounded executions that extend β in which a $rcv(m)_j$ occurs by the end of Decay phase $g+\phi$. Then by Lemma 5.5, Conclusion 1, $Pr_\beta(B'|A) \geq 1-\epsilon$.
Since Decay phase $g+\phi$ ends at time $(g+\phi)\sigma$ and $t \geq (g-1)\sigma$ by choice of g, we have that Decay phase $g+\phi$ ends by time $\leq t+(\phi+1)\sigma$. It follows that $Pr_\beta(B|A) \geq 1-\epsilon$, as needed for the receive delay bound.
Similarly, Lemma 5.6, Conclusion 1, implies that probabilistic execution satisfies the second probabilistic requirement, on the acknowledgement delay bound. Here, we use $f_{ack} = (\phi+1)\sigma$ and $\epsilon_{ack} = \epsilon\Delta$.
Finally, Lemma 5.4, Conclusion 1, implies that the probabilistic execution satisfies the progress delay bound. Here, we use $f_{prog} = (h+1)\sigma$ and $\epsilon_{prog} = (\frac{7}{8})^h$. □

Corollary 5.8. *$DMAC(\phi)$ implements the probabilistic abstract MAC layer with $f_{rcv} = f_{ack} = O(\Delta \log(\frac{1}{\epsilon}) \log(\Delta))$, $f_{prog} = O(h \log(\Delta))$, $\epsilon_{rcv} = \epsilon$, $\epsilon_{ack} = \epsilon\Delta$, $\epsilon_{prog} = (\frac{7}{8})^h$, and $t_{abort} = O(1)$.*

5.5. Implementing the Basic Layer

In this section, we prove a theorem saying that, with probability $\geq 1-\epsilon$, algorithm $DMAC(\phi)$ implements the basic abstract MAC layer, for certain values of ϵ and ϕ. Actually, our theorem doesn't quite say this. Our general definition of implementation for an abstract MAC layer says that the layer's guarantees should hold when the implementation is combined with an *arbitrary* probabilistic environment *Env*. Here, we show that the guarantees hold when the implementation is combined with an *Env* satisfying a constraint, namely, that in any execution, *Env* submits at most b *bcast*s, for some fixed positive integer b. Note that this constraint implies that the total number of external MAC layer events (*bcast*, *ack*, *abort*, and *rcv*) is at most $b(\Delta+2)$. For this section, we fix constants:

- ϵ, a real number, $0 < \epsilon \leq 1$.
- b, a positive integer. This bounds the number of *bcast* events.
- $a = b(\Delta+2)$. This bounds the total number of external MAC layer events.
- $\epsilon_1 = \frac{\epsilon}{2a}$. This is a smaller error probability, which we will use to bound errors for some auxiliary properties.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1}) \rceil$. This is the number of Decay phases we will consider for the receive and acknowledgement bounds.

- $h = \log_{\frac{8}{7}}(\frac{1}{\epsilon_1})$, the real number such that $(\frac{7}{8})^h = \epsilon_1$.

We define the four parameters for the basic abstract MAC layer:

- $f_{rcv} = f_{ack} = (\phi + 1)\sigma$.
- $f_{prog} = (\lceil h \rceil + 1)\sigma$.
- $t_{abort} = 1$.

Before stating the theorem, we define some terminology for describing violations of correctness conditions. First, we define the set AV, which represents the executions in which the acknowledgement delay bound is violated. We express AV as the union of sets AV_q, each of which describes a violation starting from the q^{th} *bcast* event.

Definition 5.9 (AV_q, where q is a positive integer, $1 \leq q \leq b$)**.** *If α is a time-unbounded execution, then we say that $\alpha \in AV_q$ provided that at least q bcast events occur in α and the following holds. Let $bcast(m)_i$ be the q^{th} bcast event. Then $ack(m)_i$ occurs in α, and for some neighbor j of i, a $rcv(m)_j$ does not precede the $ack(m)_i$. We define $AV = \bigcup_{1 \leq q \leq b} AV_q$.*

Next, we define the set PV, which represents the executions in which the progress delay bound is violated.

Definition 5.10 (PV)**.** *If α is a time-unbounded execution, then we say that $\alpha \in PV$ provided that there is a closed prefix β of α such that the following holds. Let t be the ending time of β. 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. Then I is nonempty, no $abort(m_i)_i$ occurs in α for any $i \in I$, no rcv_j occurs by time $t + f_{prog}$ for any m_i, $i \in I$, nor for any message whose bcast occurs after β, and, for some $i \in I$, $ack(m_i)_i$ does not occur by time $t + f_{prog}$.*

We can express PV as the union of sets WPV_q, where WPV_q describes a violation starting from the q^{th} external MAC layer event:

Definition 5.11 (WPV_q, where q is a positive integer, $1 \leq q \leq a$)**.** *If α is a time-unbounded execution, then we say that $\alpha \in WPV_q$ provided that at least q external MAC layer events occur in α, β is the closed prefix of α ending with the q^{th} such event, and the following holds. Let t be the ending time of β. 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. Then I is nonempty, no $abort(m_i)_i$ occurs in α for any $i \in I$, no rcv_j occurs by time $t + f_{prog}$ for any m_i, $i \in I$, nor for any message whose bcast occurs after β, and, for some $i \in I$, $ack(m_i)_i$ does not occur by time $t + f_{prog}$. We define $WPV = \bigcup_{1 \leq q \leq a} WPV_q$.*

Lemma 5.12. *$PV = WPV$.*

Proof. Clearly $WPV \subseteq PV$; we argue that $PV \subseteq WPV$. Suppose that $\alpha \in PV$. Then by definition of PV, α is a time-unbounded execution with a closed prefix β such that the following holds. Let t be the ending time of β. 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. Then I is nonempty, no $abort(m_i)_i$ occurs in α for any $i \in I$, no rcv_j occurs by time $t + f_{prog}$ for any m_i, $i \in I$, nor for any message whose *bcast* occurs after β, and, for some $i \in I$, $ack(m_i)_i$ does not occur by time $t + f_{prog}$.
Define β' to be the prefix of β ending with the last external MAC event in β. We know that some such event exists, because some neighbor of j has an active *bcast* at the end of β. Let $t' \leq t$ be the ending time of β'. Let I' be the set of neighbors of j that have active *bcast*s at the end of β'; since no external MAC events occur in β after β', we have $I' = I$. Since no rcv_j occurs by time $t + f_{prog}$ for any m_i, $i \in I$, nor for any message whose *bcast* occurs after β, we have that no rcv_j occurs by time $t' + f_{prog} \leq t + f_{prog}$ for any m_i nor for any message whose *bcast* occurs after β'. Since for some $i \in I$, $ack(m_i)_i$ does not occur by time $t + f_{prog}$, it also does not occur by time $t' + f_{prog}$. Therefore, β' illustrates that $\alpha \in WPV$. □

Theorem 5.13. *Consider the system $DMAC(\phi) \| Env \| Net$, where Env is a probabilistic environment that submits at most b bcasts. Consider the unique probabilistic execution of $DMAC(\phi) \| Env \| Net$.*
Then with probability at least $1 - \epsilon$, the probabilistic execution yields an execution that satisfies all the properties of the basic abstract MAC layer, with f_{rcv}, f_{ack}, f_{prog}, and t_{abort} as defined above.

Proof. We must show that, with probability at least $1-\epsilon$, the execution satisfies all the properties that define the basic abstract MAC layer, including all correctness guarantees and delay bounds. Theorem 5.7 implies that the algorithm satisfies all the non-probabilistic properties. Also, 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 hence by time $f_{ack} = (\phi+1)\sigma$. Thus, if the implementation fails for an execution α, it must be because $\alpha \in AV \cup PV$. We show that $Pr(AV \cup PV) \leq \epsilon$.

Claim 1: $Pr(AV) \leq \frac{\epsilon}{2}$.

Proof of Claim 1: Consider any particular q, $1 \leq q \leq b$. We apply Lemma 5.6, Conclusion 2, with ϵ in that lemma instantiated as $\frac{\epsilon_1}{\Delta}$. We use the total probability theorem (see, e.g., [23]) to combine the resulting bounds for different branches of the probabilistic execution, to obtain:

$$Pr(AV_q) \leq \frac{\epsilon_1}{\Delta} \cdot \Delta = \epsilon_1 = \frac{\epsilon}{2a} \leq \frac{\epsilon}{2b}.$$

Then, using a union bound for all values of q, we obtain that

$$Pr(AV) \leq \frac{\epsilon}{2b} \cdot b = \frac{\epsilon}{2}.$$

Claim 2: $Pr(PV) \leq \frac{\epsilon}{2}$.

Proof of Claim 2: Consider any particular q, $1 \leq q \leq a$. We apply Lemma 5.4, Conclusion 2, with h in that lemma instantiated as our $\lceil h \rceil$, and use the total probability theorem to combine the bounds for different branches of the probabilistic execution, to obtain:

$$Pr(WPV_q) \leq \left(\frac{7}{8}\right)^h = \epsilon_1 \leq \frac{\epsilon}{2a}.$$

Then, using a union bound for all values of q, we obtain that

$$Pr(WPV) \leq \frac{\epsilon}{2a} \cdot a = \frac{\epsilon}{2}.$$

In view of Lemma 5.12, we have:

$$Pr(PV) \leq \frac{\epsilon}{2}.$$

By Claims 1 and 2, $Pr(AV \cup PV) \leq \epsilon$, as needed. □

Corollary 5.14. *Consider the system $DMAC(\phi) \| Env \| Net$, where Env is a probabilistic environment that submits at most b bcasts. Consider the unique probabilistic execution of $DMAC(\phi) \| Env \| Net$.*
Then with probability at least $1-\epsilon$, the execution satisfies all the properties of the basic abstract MAC layer, 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 $t_{abort} = O(1)$.

6. Global Broadcast

So far, we have defined our basic and probabilistic abstract MAC layers, presented the *DMAC* algorithm, and proved that *DMAC* implements both layers. This completes the first part of our work. Now we turn to the second part: defining and analyzing single-message and multi-message global broadcast protocols over the MAC layers. In this section, we define the global broadcast problem and present the broadcast protocols that we will consider.

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 at the beginning of execution; however, for this case, for consistency with prior literature, we assume that the message starts out in process i_0's state.

Our broadcast algorithms are simple greedy algorithms, based on the Basic Multi-Message Broadcast (*BMMB*) algorithm of [1, 2]. These algorithms are intended to be combined with a (basic or probabilistic) MAC layer.

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*.

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 [4]. *BMMB*-Decay(ϕ) consists of *BMMB* composed with $DMAC(\phi)$.

7. Analysis of the Single-Message Broadcast Algorithm

In this section, we analyze the *BSMB*-Decay(ϕ) single-message global broadcast protocol using both abstract MAC layers. In Section 7.1, we consider the basic layer and in Section 7.2, the probabilistic layer. We use different values of ϕ in these two subsections.

We carry out this analysis by combining results from Section 5 for our MAC layer implementation with higher-level analysis of the global broadcast algorithm. Our theorems, Theorems 7.1 and 7.8, 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 some time t. The goal is to minimize t, as a function of ϵ and various graph parameters. The analysis using the basic MAC 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 [4]. The analysis using the probabilistic MAC layer yields the same bounds as in [4], but the high-level analysis of *BSMB* over the MAC layer must be redone. The ideas in this analysis are derived from those in the portion of the analysis of [4] 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 and reusable pieces.

7.1. Analysis Using Basic Abstract MAC

In this subsection, 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 $1 - \epsilon$. In this section, when we talk about "executions", we mean executions of *BSMB*-Decay(ϕ) together with our physical network and a probabilistic environment, that is, of *BSMB*-Decay(ϕ)$\|Env\|Net$.

To define ϕ (the number of Decay phases), we define constants:

- $b = n$. This is a bound on the number of *bcast* events. In this algorithm, the single message gets *bcast* at most once by each process.
- $a = n(\Delta + 2)$. This is a bound on the total number of external MAC layer events.
- $\epsilon_1 = \frac{\epsilon}{2a}$.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1}) \rceil$.

Theorem 7.1. *The* *BSMB*-Decay(ϕ) *algorithm guarantees that, with probability at least* $1 - \epsilon$, *rcv events, and hence, deliver events, occur at all nodes* $\neq i_0$ *by time*

$$O(D \log(\frac{n}{\epsilon}) \log(\Delta)).$$

Proof. Theorem 3.2 of [2] implies that when the $BSMB$ algorithm is used together with the basic abstract MAC layer, the message is always received everywhere within time $O(Df_{prog})$. Based on the constants defined in Section 5.5, and using the assumption that $\sigma = \lceil \log(\Delta + 1) \rceil$, we substitute

$$f_{prog} = O(h \log(\Delta)), h = O(\log(\frac{1}{\epsilon_1})), \epsilon_1 = \frac{\epsilon}{2a}, \text{ and } a = O(n\Delta),$$

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.
However, instead of the basic abstract MAC layer, we have an algorithm that implements the abstract MAC with probability at least $1 - \epsilon$, whenever it is placed in an environment that submits at most n *bcast*s. Since this is true for the environment consisting of the *BSMB* protocol (plus its own environment), Theorem 5.13 implies that, with probability at least $1 - \epsilon$, the MAC layer achieves the progress bound f_{prog} for every message. That implies that the entire system achieves the required message delivery bound with probability at least $1 - \epsilon$. □

7.2. Analysis Using Probabilistic Abstract MAC

In this section, we use our probabilistic MAC layer to improve the bound of Section 7.1 to $O((D + \log(\frac{n}{\epsilon})) \log(\Delta))$. This is the same bound as in [4], and our analysis uses similar ideas. However, we have split the analysis into two parts using an abstract MAC layer.
In our analysis, we first assume a probabilistic abstract MAC layer with parameters f_{prog}, f_{ack}, ϵ_{prog}, and ϵ_{ack} and analyze the complexity of *BSMB* in terms of those parameters. Then, in Section 8.3.4, we replace the abstract layer with *DMAC* and combine our bounds for *DMAC* with our bounds for *BSMB* to obtain our overall result, Theorem 7.8.
In Section 7.2.2, 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. In Section 7.2.3, we consider the system $BSMB\text{-}\mathrm{Decay}(\phi) \| Env \| Net$, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$, and *Env* is some probabilistic environment.

7.2.1. Progress Conditions

We define two constants:

$$\gamma_3 = \frac{3}{1 - \epsilon_{prog}} \quad \text{and} \quad \gamma_2 = \frac{2}{1 - \epsilon_{prog}}. \tag{5}$$

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

Definition 7.2 ($PC_j^\delta(\tau)$, where $j \in V - \{i_0\}$ and δ and τ are nonnegative reals)**.** *We say that $\alpha \in PC_j^\delta(\tau)$ if a rcv_j event occurs in α by time*

$$(\gamma_3 dist(i_0, j) + \gamma_2 \tau)(f_{prog} + \delta).$$

Also, we define:

$$PC^\delta(\tau) = \bigcap_j PC_j^\delta(\tau).$$

Let $PC_j(\tau)$ and $PC(\tau)$ denote $PC_j^0(\tau)$ and $PC^0(\tau)$, respectively.

7.2.2. Probabilistic Upper Bound on Message Delivery Time

In this section, we prove a lower bound on the probability that the message is delivered everywhere, within a certain time bound that depends on the diameter of the network and on f_{prog}. 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 nonzero δ suffices to handle the race conditions.

Lemma 7.3. *Let τ be a nonnegative real number, $j \in V - \{i_0\}$. Let δ be a positive real. Then*

$$Pr(PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof. We begin with an overview of the proof. 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 (in Claim 2) 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 (Claim 3) 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 (Claim 4), the probability that $X_0 = 1$ is at least $1 - \epsilon_{prog}$. We then apply Lemma 2.3 (where the Y_q are 1 with probability exactly $1 - \epsilon$) to obtain the final bound.

Now we give the details. Write $P_{i_0,j}$ as $i_0, i_1, i_2, \ldots, i_d = j$. Define $t_q = q(f_{prog} + \delta)$ for every nonnegative integer q. Let the random variable $Dist_q$ be the maximum l, $1 \leq l \leq d$, such that a rcv_{i_l} event occurs by time t_q; if no such event occurs then define $Dist_q = 0$. Then $Dist_q$ is well-defined for each execution and we have

$$\forall q \geq 0, Dist_q \geq 0. \tag{6}$$

Also, by definition of $Dist_q$,

$$\forall q \geq 0 : \quad Dist_{q+1} \geq Dist_q. \tag{7}$$

Define a 0-1 random variable X_q, $q \geq 0$, by

$$X_q = \begin{cases} 1 & \text{if the execution is in } \bar{N}, \\ 1 & \text{if } Dist_q = d, and \\ \min(1, Dist_{q+1} - Dist_q) & \text{otherwise.} \end{cases} \tag{8}$$

Claim 1: For every time-unbounded execution α and for every $r \geq 1$, if α satisfies $\sum_{q=0}^{r-1} X_q \geq d$ then either α satisfies $Dist_r = d$ or $\alpha \in \bar{N}$.

Proof of Claim 1: By contradiction. Suppose that α satisfies $\sum_{q=0}^{r-1} X_q \geq d$, α does not satisfy $Dist_r = d$ and $\alpha \in N$. Then (7) implies that it is not the case that α satisfies $Dist_q = d$ for any q, $0 \leq q \leq r-1$. Consequently, all X_q, $0 \leq q \leq r-1$, are determined using Case 3 of (8). Then α satisfies:

$$Dist_r - Dist_0 = \sum_{q=0}^{r-1} (Dist_{q+1} - Dist_q) \geq \sum_{q=0}^{r-1} X_q \geq d.$$

Thus, α satisfies $Dist_r \geq Dist_0 + d$, so by (6) and the fact that $Dist_r \leq d$, we get that α satisfies $Dist_r = d$, a contradiction.

Claim 1 implies that

$$\forall r \geq 1 : Pr((Dist_r = d) \cup \bar{N}) \geq Pr(\sum_{q=0}^{r-1} X_q \geq d). \tag{9}$$

Claim 2: Let α be a time-unbounded execution and $q \geq 0$. Let β be any finite prefix of α that ends at time $t_q + \delta \leq t_{q+1}$. Then $Pr_\beta(X_q = 1) \geq 1 - \epsilon_{prog}$.

Proof of Claim 2: Note that the values of random variables $Dist_0, \ldots, Dist_q$ and $X_1, \ldots, X_{q-1}$ for all $\alpha \in A_\beta$ are determined solely by the prefix β. (The notation A_β is defined in Section 2.3.) So we will sometimes refer to the values of these variables in β.
If β contains any *ack* events without all corresponding *rcv* events, then $A_\beta \subseteq \bar{N}$. Then by Case 1 of (8), we get $X_q = 1$ in β, so $Pr_\beta(X_q = 1) = 1$, which suffices. So from now on, assume that every *ack* event in β is preceded by all corresponding *rcv* events.
If $Dist_q = d$ in β, then by Case 2 of (8), we get $X_q = 1$ in β, so again $Pr_\beta(X_q = 1) = 1$. So assume that $Dist_q = e$ in β, where $0 \leq e < d$.
If $e = 0$, then a $bcast_{i_0}$ occurs at time $t_0 = 0$, so $bcast_{i_0}$ occurs in β. If $e > 0$, then by the definition of $Dist_q$, a rcv_{i_e} event occurs by time t_q, which implies that a $bcast_{i_e}$ occurs in β. In either case, a $bcast_{i_e}$ occurs in β.
If ack_{i_e} occurs in β, then, by assumption, it must be preceded by a $rcv_{i_{e+1}}$. Then by Case 3 of (8), we again have $Pr_\beta(X_q = 1) = 1$. So assume that ack_{i_e} does not occur in β.
Let J be the set of neighbors of i_{e+1} that have an active *bcast* at the end of β. Then J is nonempty because ack_{i_e} does not occur in β and the *BSMB* protocol does not use *abort* events. If any of these active $bcast(m'')$ events causes a $rcv_{i_{e+1}}$ in β, then by Case 3 of (8), we have $Pr_\beta(X_q = 1) = 1$. So assume that none of these active *bcast* events causes a $rcv_{i_{e+1}}$ in β.
Then by the definition of f_{prog}, applied to β and node i_{e+1}, with probability at least $1 - \epsilon_{prog}$ (according to Pr_β), either a $rcv_{i_{e+1}}$ occurs by time $(t_q + \delta) + f_{prog} = t_{q+1}$, or else an ack_{i_e} occurs by time t_{q+1} with no preceding $rcv_{i_{e+1}}$. In either case, we claim that $X_q = 1$ in the probabilistically-chosen execution: If a $rcv_{i_{e+1}}$ occurs by time t_{q+1}, then this follows from Case 3 of (8). On the other hand, if an ack_{i_e} occurs by time t_{q+1} with no preceding $rcv_{i_{e+1}}$, then the execution is in $\bar{N}$, so this follows from Case 1 of (8). Thus, we have $Pr_\beta(X_q = 1) \geq 1 - \epsilon_{prog}$.

Claim 3: For every $q \geq 1$ and every $x_0, x_1, ..., x_{q-1} \in \{0, 1\}$,

$$Pr(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \geq 1 - \epsilon_{prog}.$$

Proof of Claim 3: Fix q, $x_0, \ldots, x_{q-1}$. Let $\mathcal{B}$ be the set of finite prefixes β of time-unbounded executions α such that β ends at time $t_q + \delta$, and in which

$$\forall i, 0 \leq i \leq q-1 : \quad X_i = x_i.$$

Let $\mathcal{C}$ be the set of minimal elements of $\mathcal{B}$, that is, $\mathcal{C} = \{\beta \in \mathcal{B} | \ \nexists \beta' \in \mathcal{B}$ such that β' is a proper prefix of $\beta\}$. Note that every time-unbounded execution α in which

$$\forall i, 0 \leq i \leq q-1 : \quad X_i = x_i,$$

is in exactly one set of the form A_β for $\beta \in \mathcal{C}$.
Using Claim 2, we get

$$\begin{aligned}
&Pr(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta \in \mathcal{C}} Pr(X_q = 1 | A_\beta \wedge X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \cdot Pr(A_\beta | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta \in \mathcal{C}} Pr(X_q = 1 | A_\beta) \cdot Pr(A_\beta | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta \in \mathcal{C}} Pr_\beta(X_q = 1) \cdot Pr(A_\beta | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&\geq \sum_{\beta \in \mathcal{C}} (1 - \epsilon_{prog}) Pr(A_\beta | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= (1 - \epsilon_{prog}) \sum_{\beta \in \mathcal{C}} Pr(A_\beta | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= (1 - \epsilon_{prog}).
\end{aligned}$$

Claim 4:

$$Pr(X_0 = 1) \geq 1 - \epsilon_{prog}.$$

Proof of Claim 4: The proof is similar to that for Claim 3, but simpler. Let $\mathcal{B}$ be the set of finite prefixes β of time-unbounded executions such that β ends at time δ. Let $\mathcal{C}$ be the set of minimal elements of $\mathcal{B}$, Note that every time-unbounded execution α is in exactly one set of the form A_β for $\beta \in \mathcal{C}$.
Using Claim 2, we get

$$\begin{aligned} Pr(X_0 = 1) &= \sum_{\beta \in \mathcal{C}} Pr(X_0 = 1 | A_\beta) Pr(A_\beta) \\ &= \sum_{\beta \in \mathcal{C}} Pr_\beta(X_0 = 1) Pr(A_\beta) \\ &\geq \sum_{\beta \in \mathcal{C}} (1 - \epsilon_{prog}) Pr(A_\beta) \\ &= (1 - \epsilon_{prog}) \sum_{\beta \in \mathcal{C}} Pr(A_\beta) \\ &= (1 - \epsilon_{prog}). \end{aligned}$$

We now return to the main proof. Let Y_q, $0 \leq q$, be a collection of independent 0-1 random variables such that

$$Pr(Y_q = 1) = 1 - \epsilon_{prog}.$$

By Claim 3, we have that for every $q \geq 1$, and for every $x_0, x_1, ..., x_{q-1} \in \{0, 1\}$,

$$Pr(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \geq Pr(Y_q = 1).$$

By Claim 4, we have that

$$Pr(X_0 = 1) \geq Pr(Y_0 = 1).$$

It follows from Lemma 2.2 that, for any $r \geq 1$,

$$Pr(\sum_{q=0}^{r-1} X_q \geq d) \geq Pr(\sum_{q=0}^{r-1} Y_q \geq d).$$

Therefore, by (9), we get

$$\begin{aligned} Pr((Dist_r = d) \cup \bar{N}) &\geq Pr(\sum_{q=0}^{r-1} Y_q \geq d) \\ &= 1 - Pr(\sum_{q=0}^{r-1} Y_q < d). \end{aligned} \tag{10}$$

Now we set $r = \lfloor \gamma_3 d + \gamma_2 \tau \rfloor$. By the definition of PC_j^δ, we have that, for any time-unbounded execution α, if $Dist_r = d$ in α, then $\alpha \in PC_j^\delta(\tau)$. Hence, by (10), we have

$$Pr(PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - Pr(\sum_{q=0}^{r-1} Y_q < d). \tag{11}$$

Now we apply Lemma 2.3, with $p = 1 - \epsilon_{prog}$, to obtain an upper bound for the probability of the sum on the right-hand side of (11):

$$Pr(\sum_{q=0}^{r-1} Y_q < d) \leq e^{-\tau}. \tag{12}$$

Then by (11) and (12), we get

$$Pr(PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - e^{-\tau},$$

which completes the proof. □

Note that the δ term is used in the proof of Lemma 7.3 to ensure 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 7.4. *Let τ be a nonnegative real number, and let $j \in V - \{i_0\}$. Then*

$$Pr(PC_j(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof. Follows since Lemma 7.3 holds for every $\delta > 0$. In detail, Lemma 7.3 says that, for every $\delta > 0$. $Pr(PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}$. Note that, for $0 < \delta_1 \leq \delta_2$, we have $PC_j^{\delta_1}(\tau) \cup \bar{N} \subseteq PC_j^{\delta_2}(\tau) \cup \bar{N}$. Therefore,

$$Pr(\bigcap_{\delta>0} PC_j^\delta(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}. \tag{13}$$

We claim that

$$\bigcap_{\delta>0} PC_j^\delta(\tau) \cup \bar{N} = PC_j(\tau) \cup \bar{N}. \tag{14}$$

One direction is obvious; we argue the other, that

$$\bigcap_{\delta>0} PC_j^\delta(\tau) \cup \bar{N} \subseteq PC_j(\tau) \cup \bar{N}.$$

So, let $\alpha \in \bigcap_{\delta>0} PC_j^\delta(\tau) \cup \bar{N}$. If $\alpha \in \bar{N}$ then $\alpha \in PC_j(\tau) \cup \bar{N}$ and we are done. On the other hand, if $\alpha \in \bigcap_{\delta>0} PC_j^\delta(\tau)$, then for every $\delta > 0$, α contains a rcv_j event at a time that is $\leq (\gamma_3 d + \gamma_2 \tau)(f_{prog} + \delta)$. Since α cannot contain an infinite sequence of discrete events at successively decreasing times (a basic property of timed executions for PTIOAs), the only possibility is that α contains a rcv_j event at a time that is $\leq (\gamma_3 d + \gamma_2 \tau) f_{prog}$. Thus, $\alpha \in PC_j(\tau)$, which suffices.
Then by (13) and (14), we get that

$$Pr(PC_j(\tau) \cup \bar{N}) \geq 1 - e^{-\tau},$$

as needed. □

Lemma 7.4 allows us to bound the probability of the progress condition for all nodes.

Lemma 7.5. *Let τ be a nonnegative real number. Then*

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

Proof. By definition of PC, we have:

$$PC(\tau) = \bigcap_{j \neq i_0} PC_j(\tau).$$

Using a union bound and Lemma 7.4, we obtain:

$$Pr(PC(\tau) \cup \bar{N}) = Pr(\bigcap_{j \neq i_0} (PC_j(\tau) \cup \bar{N})) \geq 1 - ne^{-\tau}.$$

□

Lemma 7.5 yields a bound for the set of executions that satisfy the progress condition for all nodes, and also are nice, as defined in Section 4.3.

Lemma 7.6. *Let τ be a nonnegative real number. Then*

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

Proof. Using Lemma 7.5, we obtain:

$$\begin{aligned} Pr(PC(\tau) \cap N) &\geq Pr((PC(\tau) \cap N) \cup \bar{N}) - Pr(\bar{N}) \\ &= Pr(PC(\tau) \cup \bar{N}) - Pr(\bar{N}) \\ &\geq 1 - ne^{-\tau} - Pr(\bar{N}). \end{aligned}$$

□

Now we combine Lemma 7.6 with Lemma 4.2 (our upper bound on the probability of $\bar{N}$) to obtain our bound for *BSMB* over the probabilistic MAC layer.

Theorem 7.7. *Let ϵ be a real number, $0 < \epsilon \leq 1$. The BSMB protocol guarantees that, with probability at least*

$$1 - \epsilon - n\epsilon_{ack},$$

rcv events, and hence, deliver events, occur at all nodes $\neq i_0$ by time

$$(\gamma_3 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}.$$

Proof. By Lemmas 7.6 and 4.2, with probability at least

$$1 - ne^{-\tau} - n\epsilon_{ack},$$

rcv events occur at all nodes $\neq i_0$ by time

$$(\gamma_3 D + \gamma_2 \tau)f_{prog}.$$

The conclusion follows by replacing τ with $\ln(\frac{n}{\epsilon})$. □

7.2.3. Analysis of the Complete Algorithm

Finally, we combine our bound for the *BSMB* protocol in terms of the probabilistic abstract MAC layer (Theorem 7.7) with our results for *DMAC* to obtain a bound for the combined *BSMB*-Decay algorithm.

Theorem 7.8. *Let ϵ be a real number, $0 < \epsilon \leq 1$. Let $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$. The BSMB-*Decay($\phi$) *algorithm guarantees that, with probability at least $1 - \epsilon$, rcv events, and hence, 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 7.7, 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

$$(\gamma_3 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}.$$

Using the definitions of parameters for the implementation of the probabilistic layer, in Section 5.4, we may assume that $\epsilon_{prog} \leq \frac{7}{8}$, so this expression is

$$O((D + \log(\frac{n}{\epsilon}))f_{prog}).$$

Again using those 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 the *BMMB*-Decay(ϕ) multi-message global broadcast protocol using both abstract MAC layers. In Section 8.2, we consider the basic layer, and in Section 8.3, the probabilistic layer. We use different values of ϕ in these two subsections.

We carry out this analysis by combining results from Section 5 with higher-level analysis of the global broadcast algorithm. Our theorems, Theorems 8.4 and 8.21, 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.

As for single-message broadcast, 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 many ideas from Section 7.2, plus new ideas to cope with the on-line arrival of messages. It also uses a new path decomposition trick. The analysis is not easy; in fact, we believe it would be infeasible without such a decomposition.

8.1. Definitions

We define the set of broadcast messages that are concurrent with a given broadcast message m. For this, it is useful to identify the final MAC-layer *ack* event associated with each broadcast message.

Definition 8.1 (*Clear* events)**.** *Let α be an execution in N (the set of nice executions, as defined in Section 4.3), and let $m \in M$ be a message such that an arrive(m) event occurs in α. We define the event clear(m) to be the final ack(m) event in α.*

Definition 8.2 (The Set $K(m)$)**.** *Let α be an execution in N and let $m \in M$ be a message such that arrive(m) occurs in α. We 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.*

Since a broadcast message can first arrive at a node via either an *arrive* or *rcv* event, we use the following notation to combine the two possibilities:

Definition 8.3 (*get* events)**.** *A get(m)$_j$ event is defined 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.*

8.2. Analysis Using Basic Abstract MAC

In this subsection, 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 $O((D + k'\Delta)\log(\frac{n}{\epsilon})\log(\Delta))$. In this section, when we talk about "executions", we mean executions of *BMMB*-Decay(ϕ) together with our physical network and a probabilistic environment, that is, of *BMMB*-Decay(ϕ)$\|Env\|Net$.

We define constants:

- $b = kn$. This is a bound on the number of *bcast* events. In this algorithm, each of the k messages gets *bcast* at most once by each node.
- $a = kn(\Delta + 2)$. This is a bound on the total number of external MAC layer events.
- $\epsilon_1 = \frac{\epsilon}{2a}$.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1})\rceil$.

Theorem 8.4. *Let $m \in M$. The BMMB-Decay(ϕ) algorithm 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 $|K(m)| \leq k'$. Then get(m) events, and hence, deliver events, occur at all nodes in α within time

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

of the time of the arrive(m) event.

Note that if k is polynomial in n, the bound reduces to $O((D + k'\Delta)\log(\frac{n}{\epsilon})\log(\Delta))$.

Proof. Theorem 3.2 of [2] implies that when the $BMMB$ algorithm is used together with the basic abstract MAC layer, the message is always 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 5.5, we substitute

$$f_{prog} = O(\log(\frac{1}{\epsilon_1})\log(\Delta)), f_{ack} = O(\Delta\log(\frac{\Delta}{\epsilon_1})\log(\Delta)), \epsilon_1 = \frac{\epsilon}{2a}, \text{ and } a = O(kn\Delta),$$

to obtain a bound of the form

$$O((D + k')\log(\frac{nk}{\epsilon})\log(\Delta)) + (k' - 1)O(\Delta\log(\frac{nk}{\epsilon})\log(\Delta)) = O((D + k'\Delta)\log(\frac{nk}{\epsilon})\log(\Delta)).$$

Thus, if the algorithm ran with a basic abstract MAC layer with f_{prog} and f_{ack} as above, it would, in every execution, deliver each message m everywhere by the indicated bound.
However, instead of the basic abstract MAC layer, we have an algorithm that implements it with probability at least $1-\epsilon$, whenever it is placed in an environment that submits at most kn *bcast*s. Since this is true for the environment consisting of the $BMMB$ protocol (plus its own environment), Theorem 5.13 implies that, with probability at least $1 - \epsilon$, the MAC layer achieves the progress bound f_{prog} and the acknowledgment bound f_{ack}. That implies that the entire system achieves the required message delivery bounds with probability at least $1 - \epsilon$. □

8.3. Analysis Using Probabilistic Abstract MAC

Now we use our probabilistic MAC layer to improve the bound of Section 8.2 to $O((D+k'\Delta\log(\frac{nk}{\epsilon}))\log(\Delta))$. In our analysis, 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, in Section 8.3.4, we replace the abstract layer with $DMAC$ and combine the bounds for $DMAC$ and $BMMB$ to obtain Theorem 8.21.
For our analysis of $BMMB$ over the abstract MAC layer, we begin by redefining the progress condition PC, in Section 8.3.1. Then, in Section 8.3.2, 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 4.3. Finally, in Section 8.3.3, we bound the probability that an execution is well-behaved and use this to infer our probabilistic bound on message delivery time.
In Section 8.3.3, our probabilistic statements are with respect to the system $BMMB\|Mac\|Env\|Net$, where Mac is an arbitrary implementation of the abstract probabilistic MAC layer with parameters $f_{prog}, f_{ack}, \epsilon_{prog}$, and ϵ_{ack}, and Env is some probabilistic environment that submits at most k messages. In Section 8.3.4, we consider the system $BSMB$-Decay(ϕ)$\|Env\|Net$, where $\phi = \lceil 8\Delta\ln(\frac{1}{\epsilon})\rceil$, and Env is some probabilistic environment that submits at most k messages.

8.3.1. Progress Conditions

Our first definition identifies the messages whose processing is completed at a particular node or set of nodes by a designated real time:

Definition 8.5. *For any $i \in V$, nonnegative real number t and execution α, define $C_i^\alpha(t)$ to be the set of messages m such that $ack(m)_i$ occurs by time t in α.*
For any $I \subseteq V$, nonnegative real number t and execution α, define $C_I^\alpha(t)$ to be the set of messages $\bigcap_{i\in I} C_i^\alpha(t)$, that is, the set of messages m such that $ack(m)_i$ occurs by time t for every $i \in I$.

We now redefine the progress condition PC. In this definition, we use the same constants $\gamma_3 = \frac{3}{1-\epsilon_{prog}}$ and $\gamma_2 = \frac{2}{1-\epsilon_{prog}}$ defined in Section 7.2.1. As before, the progress condition is parameterized by a nonnegative real τ.
The condition refers to two arbitrary nodes i and j. It says that, if a broadcast message is received by, or arrives at, node i by time t, and is "new" in that it is not already completely processed in the neighborhood of the path from i to j by time t, then either this message or some other "new" message is received by, or arrives at node j within a certain amount of time.

Definition 8.6 (Progress Condition $PC_{i,j}(\tau)$, where $i, j \in V, i \neq j$, and τ is a nonnegative real)**.** *Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$ (note that I does not include node $i = i_0$). 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 $m \notin C^\alpha_{\Gamma(I)}(t)$ occurs in α by time t, then a $get(m')_j$ event for some message $m' \notin C^\alpha_{\Gamma(I)}(t)$ occurs by time

$$t + (\gamma_3 d + \gamma_2 \tau) f_{prog}.$$

Also, we define:

$$PC(\tau) = \bigcap_{i,j,i\neq j} PC_{i,j}(\tau).$$

Now we define an alternative progress condition $WPC_{i,j}$. $WPC_{i,j}$ differs from $PC_{i,j}$ in that it is stated in terms of real times at which *get* or *ack* events occur, rather than arbitrary real times. We prove that $WPC_{i,j}$ is in fact equivalent to $PC_{i,j}$. $WPC_{i,j}$ is more convenient to use in a union bound analysis in Section 8.3.3.

Definition 8.7 (The set of executions $WPC_{i,j}(\tau)$, where $i, j \in V$, $i \neq j$, and τ is a nonnegative real)**.** *Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$. An execution α is in $WPC_{i,j}$ if for every nonnegative real t, the following holds:*
If a get or ack event occurs anywhere 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

$$t + (\gamma_3 d + \gamma_2 \tau) f_{prog}.$$

Also, we define:

$$WPC(\tau) = \bigcap_{i,j,i\neq j} WPC_{i,j}(\tau).$$

Lemma 8.8. *For every $i, j \in V$, $i \neq j$, and nonnegative real τ:*

$$PC_{i,j}(\tau) = WPC_{i,j}(\tau).$$

Proof. Fix i, j, and τ. Assume that $P_{i,j} : i = i_0, i_1, \ldots, i_d = j$ and $I = \{i_1, \ldots, i_d\}$. We show that $PC_{i,j}(\tau) \subseteq WPC_{i,j}(\tau)$ and $WPC_{i,j}(\tau) \subseteq PC_{i,j}(\tau)$.

1. $PC_{i,j}(\tau) \subseteq WPC_{i,j}(\tau)$.

   Let α be any execution in $PC_{i,j}(\tau)$; we show that $\alpha \in WPC_{i,j}(\tau)$. Fix a nonnegative real t, and suppose that a *get* or *ack* happens at time t. Further, suppose that a $get(m)_i$ event occurs for some message $m \notin C^\alpha_{\Gamma(I)}(t)$ by time t. By the definition of $PC_{i,j}(\tau)$, a $get(m')_j$ event for some message $m' \notin C^\alpha_{\Gamma(I)}(t)$ occurs in α by time $t + (\gamma_3 d + \gamma_2 \tau) f_{prog}$. It follows that $\alpha \in WPC_{i,j}(\tau)$.

2. $WPC_{i,j}(\tau) \subseteq PC_{i,j}(\tau)$.

   Let α be any execution in $WPC_{i,j}(\tau)$; we show that $\alpha \in PC_{i,j}(\tau)$. Fix t, and suppose that a $get(m)_i$ event occurs for some message $m \notin C^\alpha_{\Gamma(I)}(t)$ by time t. We show that a $get(m')_j$ event occurs for some

message $m' \notin C^{\alpha}_{\Gamma(I)}(t)$ by time $t + (\gamma_3 d + \gamma_2 \tau) f_{prog}$. Fix m to be any message such that $m \notin C^{\alpha}_{\Gamma(I)}(t)$ and a $get(m)_i$ event occurs by time t. Let t', $t' \leq t$, be the largest real number such that either a *get* or an *ack* event occurs at time t'. We have $C^{\alpha}_{\Gamma(I)}(t) \subseteq C^{\alpha}_{\Gamma(I)}(t')$, because, by definition of t', no *ack* event occurs after t' and by time t. Since $C^{\alpha}_{\Gamma(I)}(t') \subseteq C^{\alpha}_{\Gamma(I)}(t)$, we get $C^{\alpha}_{\Gamma(I)}(t) = C^{\alpha}_{\Gamma(I)}(t')$.

Also, by choice of t', the $get(m)_i$ event occurs by time t', and either a *get* or an *ack* occurs at time t'. Then by the definition of $WPC_{i,j}(\tau)$, a $get(m')_j$ event, $m' \notin C^{\alpha}_{\Gamma(I)}(t')$ occurs by time $t' + (\gamma_3 d + \gamma_2 \tau) f_{prog} \leq t + (\gamma_3 d + \gamma_2 \tau) f_{prog}$. It follows that $\alpha \in PC_{i,j}(\tau)$, as needed.

□

The following lemma follows immediately from Lemma 8.8.

Lemma 8.9. *For every nonnegative real τ:*

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

8.3.2. Message Delivery Guarantee for Well-Behaved Executions

In this subsection, we prove a lemma giving a non-probabilistic upper bound on message delivery time in "well-behaved" executions. By "well-behaved", we mean that the executions satisfy the progress condition and also are "nice"; that is, they are in $PC(\tau) \cap N$. The lemma 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 positive integer, 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 of this result is the most challenging one in the paper.

Lemma 8.10. *Let τ be a positive integer. For nonnegative integers d and l, with $l > 0$, define*

$$t_{d,l} := t_0 + ((\gamma_3 + \gamma_2)d + ((\gamma_3 + 2\gamma_2)\tau + \gamma_3 + \gamma_2)l)\, f_{prog} + (l-1) f_{ack}.$$

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 $l \geq 1$, 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. We prove the lemma by induction on l.

- *Base case: $l = 1$.*

  We consider subcases based on whether $d = 0$ or $d > 0$. If $d = 0$, then $j = i$. Let m' be the first message in i's queue immediately after the $arrive(m)_i$ event. Then $m' \in M'$, $get(m')_i$ occurs by time $t_0 \leq t_{0,1}$, and $ack(m')_i$ occurs by time $t_{0,1} + f_{ack}$, so Statement 2 is true using $M'' = \{m'\}$.

  If $d > 0$, then we use the fact that $\alpha \in PC_{i,j}(\tau)$ to obtain a message with the needed properties. Write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$ and let $I = \{i_1, \ldots, i_d\}$. If $m \in C^{\alpha}_j(t_0)$ then Statement 1 is true. If not, then $m \notin C^{\alpha}_{\Gamma(I)}(t_0)$. Then since $\alpha \in PC_{i,j}(\tau)$, a $get(m')_j$ event for some $m' \notin C^{\alpha}_{\Gamma(I)}(t_0)$ occurs by time

  $$t_0 + (\gamma_3 d + \gamma_2 \tau) f_{prog} < t_{d,1}.$$

  If m' reaches the front of j's queue by time $t_{d,1}$, then $ack(m')_j$ occurs by time $t_{d,1} + f_{ack}$. Also, note that $m' \in M'$, because $m' \notin C^{\alpha}_{\Gamma(I)}(t_0)$. So Statement 2 is true using $M'' = \{m'\}$. Otherwise, that is, if m' does not reach the front of j's queue by time $t_{d,1}$, then in the last state of α at time $t_{d,1}$, some other message m'' is first on j's queue. This implies that $get(m'')_j$ occurs by time $t_{d,1}$ and $ack(m'')_j$ occurs by time $t_{d,1} + f_{ack}$. Also, note that $m'' \in M'$ because m'' is still in j's queue at time $t_{d,1} > t_0$. So again, Statement 2 is true for j and l, in this case using $M'' = \{m''\}$.

- *Inductive step: $l > 1$, assume the lemma for $l-1$ and all values of d.*

  Now we proceed by induction on d.

  - *Base case: $d = 0$.*

    Then $j = i$. Suppose there are exactly l_0 messages in i's queue immediately after the $arrive(m)_i$ occurs at time t_0. Note that the $arrive(m)_i$ event is also the $get(m)_i$ event. All of these l_0 messages are in M', and all of their get_i events occur by time $t_0 \leq t_{0,l}$. If $l \geq l_0$ then we have that $ack(m)_i$ occurs by time $t_0 + l_0 f_{ack} \leq t_0 + l f_{ack} \leq t_{0,l} + f_{ack}$, which implies that Statement 1 is true. On the other hand, if $l < l_0$, then $ack(m')_i$ events occur for the first l messages on the queue by time $t_0 + l f_{ack} \leq t_{0,1} + f_{ack}$, so Statement 2 is true.

  - *Inductive step: $d > 1$, 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, that is, that 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.

    Since Statement 1 is false for j and l, it is also false for j and $l-1$. Then by inductive hypothesis, Statement 2 must be 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 $t_{d,l-1} + f_{ack} < t_{d,l}$. Since Statement 1 is false for j and $l-1$, we have $m \notin M''$. Fix this set M'' for the rest of the proof.

    Claim 1: If $get(m')_j$ occurs by time $t_{d,l}$ for some $m' \in M' - M''$, then Statement 2 is true for j and l.

    Proof of Claim 1: Suppose that $get(m')_j$ occurs by time $t_{d,l}$ for some particular $m' \in M' - M''$. If m' reaches the front of j's queue by time $t_{d,l}$, then $ack(m')_j$ occurs by time $t_{d,l} + f_{ack}$, so Statement 2 is true for j and l using the size l set $M'' \cup \{m'\}$. Otherwise, that is, if m' does not reach the front of j's queue by time $t_{d,l}$, then in the last state of α at time $t_{d,l}$, some other message m'' is first on j's queue. This implies that $get(m'')_j$ occurs by time $t_{d,l}$ and $ack(m'')_j$ occurs by time $t_{d,l} + f_{ack}$. Note that $m'' \in M'$ because m'' is still in j's queue at time $t_{d,l} > t_0$. Also, $m'' \notin M''$, because m'' is still in j's queue at time $t_{d,l}$ whereas ack_j events occur for all messages in M'' before that time. Then Statement 2 is true for j and l, using the size l set $M'' \cup \{m''\}$.

    Claim 2: Let j_1 and j_2 be neighbors. If $M'' \not\subseteq C^{\alpha}_{j_1}(t_{dist(i,j_1),l-1} + f_{ack})$ then for some $m' \in M' - M''$, $get(m')_{j_2}$ occurs by time $t_{dist(i,j_1),l-1} + f_{ack}$.

    Proof of Claim 2: By inductive hypothesis for j_1 and $l-1$, either Statement 1 or Statement 2 is true for j_1 and $l-1$. If Statement 1 is true then $m \in C^{\alpha}_{j_1}(t_{dist(i,j_1),l-1} + f_{ack})$. Since $\alpha \in N$, this implies that $get(m)_{j_2}$ occurs by time $t_{dist(i,j_1),l-1} + f_{ack}$, as needed. On the other hand, if Statement 2 is true, then there are at least $l-1$ elements of M' in $C^{\alpha}_{j'}(t_{dist(i,j_1),l-1} + f_{ack})$. Since $M'' \not\subseteq C^{\alpha}_{j'}(t_{dist(i,j_1),l-1} + f_{ack})$, this set must contain some message $m' \in M' - M''$. Since $\alpha \in N$, this implies that $get(m')_{j_2}$ occurs by time $t_{dist(i,j_1),l-1} + f_{ack}$, as needed.

    We return to the main proof. If for some neighbor j' of j, $M'' \not\subseteq C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack})$, then Claim 2 implies that for some $m' \in M' - M''$, a $get(m')_j$ event occurs by time $t_{dist(i,j'),l-1} + f_{ack} \leq t_{d+1,l-1} + f_{ack} < t_{d,l}$. Then Claim 1 implies that Statement 2 is true for j and l, as needed.

    The remaining case is where, for every neighbor j' of j, $M'' \subseteq C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack})$. Then 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

    $$M'' \subseteq \bigcap_{j' \in \Gamma(I_{e'})} C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack}). \tag{15}$$

    We know that e' exists because (15) holds for $e' = d-1$. For this e', we have the following property:

    Claim 3: There exists $m' \in M' - M''$ such that $get(m')_{i_{e'}}$ occurs by time $t_{e'+1,l-1} + f_{ack}$.

Proof of Claim 3: If $e' = 0$, then $m' = m$ satisfies the claim. So assume that $e' > 0$. By the way e' was chosen, there must be some neighbor j' of $i_{e'}$ such that $M'' \not\subseteq C^{\alpha}_{j'}(t_{dist(i,j'),l-1} + f_{ack})$. Then by Claim 2, for some $m' \in M' - M''$, a $get(m')_{i_{e'}}$ event occurs by time $t_{dist(i,j'),l-1} + f_{ack} \leq t_{e'+1,l-1} + f_{ack}$, as needed.

Once more, we return to the main proof. Let $d - e' = q\tau + r$, where q and r are nonnegative integers and $0 \leq r < \tau$.
First suppose that $q > 0$. By (15), we have

$$M'' \subseteq C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack}),$$

where $J = \{i_{e'+1}, \ldots i_{e'+\tau}\}$. This is because, for every $j' \in \Gamma(J)$, $dist(i, j') \leq e' + \tau + 1$.
Claim 3 says 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 m'. 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, $m' \notin C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$. In this case, 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 implies that there exists $m_1 \notin C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$ such that $get(m_1)_{i_{e'+\tau}}$ occurs by time

$$t_{e'+\tau+1,l-1} + f_{ack} + (\gamma_3\tau + \gamma_2\tau)f_{prog} \leq t_{e'+2\tau+1,l-1} + f_{ack}.$$

Note that $m_1 \in M'$, because $m_1 \notin C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$. Also, $m_1 \notin M''$, because $m_1 \notin C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$ and $M'' \subseteq C^{\alpha}_{\Gamma(J)}(t_{e'+\tau+1,l-1} + f_{ack})$. So $m_1 \in M' - M''$. 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 can repeat the same argument using the progress conditions

$$PC_{i_{e'+\tau},i_{e'+2\tau}}(\tau), PC_{i_{e'+2\tau},i_{e'+3\tau}}(\tau), \ldots PC_{i_{e'+(q-1)\tau},i_{e'+q\tau}}(\tau),$$

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 the progress condition $PC_{i_{d-r},j}(\tau)$, we show that there exists $m'' \in M' - M''$ such that $get(m'')_j$ occurs by time

$$t_{d-r+\tau+1,l-1} + f_{ack} + (\gamma_3 r + \gamma_2\tau)f_{prog} \leq t_{d,l}.$$

Now suppose that $q = 0$, that is, $d - e' < \tau$. Then using the progress condition $PC_{i_{e'},j}(\tau)$, we can show that there exists $m'' \in M' - M''$ such that $get(m'')_j$ occurs by time

$$t_{d+1,l-1} + f_{ack} + (\gamma_3 r + \gamma_2\tau)f_{prog} \leq t_{d,l}.$$

Thus, in any case, a $get(m'')_j$ event occurs for some $m'' \in M' - M''$ by time $t_{d,l}$. Then Claim 1 implies that Statement 2 is true for j and l, as needed.

□

8.3.3. Probabilistic Upper Bound on Message Delivery Time
In this section, we prove a lower bound on the probability that executions satisfy the progress condition and are nice, that is, on the probability of the event $PC(\tau) \cap N$. We then tie all the results together in Theorem 8.20.
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, the lemma gives a lower bound on the probability that either some new message is delivered to j within a short time, or else the execution is not nice. In this lemma, we consider probabilities with respect to the conditional distribution on time-unbounded executions of *BMMB* that extend a particular finite execution β. The notation A_β and Pr_β is defined in Section 2.3.
As before, the lemma includes a δ term to handle race conditions. We remove δ in a following lemma.

Lemma 8.11. *Let τ be a nonnegative real number. Consider $i, j \in V$, $i \neq j$, 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 $get(m)_i$ event occurs in β.
Let δ be a positive real. Let F^{δ} 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

$$t_0 + (\gamma_3 d + \gamma_2 \tau)(f_{prog} + \delta).$$

Then

$$Pr_{\beta}(F^{\delta} \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof. The proof follows the general outline of that for Lemma 7.3, and again uses Lemma 2.3. 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 $C^{\beta}_{\Gamma(I)}(t_0)$.
Specifically, define $t_q = t_0 + q(f_{prog} + \delta)$ for every nonnegative integer q. Let the random variable $Dist_q$ be the maximum l, $0 \leq l \leq d$, such that there exists $m' \notin C^{\beta}_{\Gamma(I)}(t_0)$ for which a $get(m')_{i_l}$ event occurs by time t_q. Since in β, and hence in all executions in A_{β}, a $get(m)_{i_0}$ event occurs by time t_0 and $m \notin C^{\beta}_{\Gamma(I)}(t_0)$, $Dist_q$ is well-defined for each execution and we have

$$\forall q \geq 0, Dist_q \geq 0. \tag{16}$$

Also, by definition of $Dist_q$,

$$\forall q \geq 0: \quad Dist_{q+1} \geq Dist_q. \tag{17}$$

Define a 0-1 random variable X_q, $q \geq 0$, by

$$X_q = \begin{cases} 1 & \text{if the execution is in } \bar{N}; \\ 1 & \text{if } Dist_q = d; \\ \min(1, Dist_{q+1} - Dist_q) & \text{otherwise.} \end{cases} \tag{18}$$

Claim 1: For every $\alpha \in A_{\beta}$ and for every $r \geq 1$, if α satisfies $\sum_{q=0}^{r-1} X_q \geq d$ then either α satisfies $Dist_r = d$ or $\alpha \in \bar{N}$.
Proof of Claim 1: By contradiction. Suppose that α satisfies $\sum_{q=0}^{r-1} X_q \geq d$, α does not satisfy $Dist_r = d$ and $\alpha \in N$. Then (7) implies that it is not the case that α satisfies $Dist_q = d$ for any q, $0 \leq q \leq r-1$. Consequently, all X_q, $0 \leq q \leq r-1$, are determined using Case 3 of (18). Then α satisfies:

$$Dist_r - Dist_0 = \sum_{q=0}^{r-1} (Dist_{q+1} - Dist_q) \geq \sum_{q=0}^{r-1} X_q \geq d.$$

Thus, α satisfies $Dist_r \geq Dist_0 + d$, so by (6) and the fact that $Dist_r \leq d$, we get that α satisfies $Dist_r = d$, a contradiction.

Claim 1 implies that

$$\forall r \geq 1 : Pr_{\beta}((Dist_r = d) \cup \bar{N}) \geq Pr(\sum_{q=0}^{r-1} X_q \geq d). \tag{19}$$

Claim 2: Let $\alpha \in A_{\beta}$ and $q \geq 0$. Let β' be any finite prefix of α that ends at time $t_q + \delta \leq t_{q+1}$. Then $Pr_{\beta'}(X_q = 1) \geq 1 - \epsilon_{prog}$.
Proof of Claim 2: Note that the values of random variables $Dist_0, \ldots, Dist_q$ and $X_1, \ldots, X_{q-1}$ for all $\alpha \in A_{\beta'}$ are determined solely by the prefix β'. So we will sometimes refer to the values of these variables in β'.

If β' contains any *ack* events without all corresponding *rcv* events, then $A_{\beta'} \subseteq \bar{N}$. Then by Case 1 of (18), we get $X_q = 1$ in β', so $Pr_{\beta'}(X_q = 1) = 1$, which suffices. So from now on, assume that every *ack* event in β' is preceded by all corresponding *rcv* events.
If $Dist_q = d$ in β', then by Case 2 of (18), we get $X_q = 1$ in β', so again $Pr_{\beta'}(X_q = 1) = 1$. So assume that $Dist_q = e$ in β', where $0 \leq e < d$.
By the definition of $Dist_q$, there exists $m_1 \notin C^{\beta}_{\Gamma(I)}(t_0)$ for which a $get(m_1)_{i_e}$ event occurs in β'. If m_1 reaches the front of i_e's queue by time t_q, then $bcast(m_1)_{i_e}$ occurs in β'. If not, then some other message m_2 is at the front of i_e's queue in the last state of β' at time t_q, in which case $bcast(m_2)_{i_e}$ occurs in β'. Note that $m_2 \notin C^{\beta}_{\Gamma(I)}(t_0)$. Thus, in either case, there exists $m' \notin C^{\beta}_{\Gamma(I)}(t_0)$ for which a a $bcast(m')_{i_e}$ event occurs in β'. Fix such m'.
If $ack(m')_{i_e}$ occurs in β', then, by assumption, it must be preceded by a $rcv(m')_{i_{e+1}}$. Then by Case 3 of (18), we again have $Pr_{\beta'}(X_q = 1) = 1$. So assume that $ack(m')_{i_e}$ does not occur in β'.
Let J be the set of neighbors of i_{e+1} that have an active $bcast(m'')$ for some message m'' at the end of β'. Then J is nonempty because $ack(m')_{i_e}$ does not occur in β' and the *BMMB* protocol does not use *abort* events. Note that for any such active $bcast(m'')$ event, we have $m'' \notin C^{\beta}_{\Gamma(I)}(t_0)$. This is because $J \subseteq \Gamma(I)$, and so all nodes in J have cleared all the messages in $C^{\beta}_{\Gamma(I)}(t_0)$ by the end of β'. If any of these active $bcast(m'')$ events causes a $rcv(m')_{i_{e+1}}$ in β', then by Case 3 of (18), we have $Pr_{\beta'}(X_q = 1) = 1$. So assume that none of these active $bcast(m'')$ events causes a $rcv(m')_{i_{e+1}}$ in β'.
Then by the definition of f_{prog}, applied to β' and node i_{e+1}, with probability at least $1 - \epsilon_{prog}$ (according to $Pr_{\beta'}$), either a $rcv(m'')_{i_{e+1}}$ occurs for some $m'' \notin C^{\beta}_{\Gamma(I)}(t_0)$ by time $(t_q + \delta) + f_{prog} = t_{q+1}$, or else an $ack(m')_{i_e}$ occurs by time t_{q+1} with no preceding $rcv(m')_{i_{e+1}}$. (For the first case, according to the definition of f_{prog}, m'' may be either a message that is active at a neighbor in J after β', or else a message whose *bcast* occurs after β'; either way, we have $m'' \notin C^{\beta}_{\Gamma(I)}(t_0)$ as claimed.) In either case, we claim that $X_q = 1$ in the probabilistically-chosen execution: If a $rcv(m'')_{i_{e+1}}$ occurs for some $m'' \notin C^{\beta}_{\Gamma(I)}(t_0)$ by time t_{q+1}, then this follows from Case 3 of (18). On the other hand, if an $ack(m')_{i_e}$ occurs by time t_{q+1} with no preceding $rcv(m')_{i_{e+1}}$, then the execution is in $\bar{N}$, so this follows from Case 1 of (18). Thus, we have: $Pr_{\beta'}(X_q = 1) \geq 1 - \epsilon_{prog}$.

Claim 3: For every $q \geq 1$ and every $x_0, x_1, ..., x_{q-1} \in \{0, 1\}$,

$$Pr_{\beta}(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \geq 1 - \epsilon_{prog}.$$

Proof of Claim 3: Fix q, $x_0, \ldots, x_{q-1}$. Let $\mathcal{B}$ be the set of finite prefixes β' of executions $\alpha \in A_\beta$ such that β' ends at time $t_q + \delta$, and in which

$$\forall i, 0 \leq i \leq q-1 : \quad X_i = x_i.$$

Let $\mathcal{C}$ be the set of minimal elements of $\mathcal{B}$, that is, $\mathcal{C} = \{\beta' \in \mathcal{B} | \ \nexists \beta'' \in \mathcal{B}$ such that β'' is a proper prefix of $\beta'\}$. Note that every $\alpha \in A_\beta$ in which

$$\forall i, 0 \leq i \leq q-1 : \quad X_i = x_i,$$

is in exactly one set of the form $A_{\beta'}$ for $\beta' \in \mathcal{C}$.

Using Claim 2, we get

$$\begin{aligned}
&Pr_\beta(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta' \in \mathcal{C}} Pr_\beta(X_q = 1 | A_{\beta'} \wedge X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \cdot Pr_\beta(A_{\beta'} | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta' \in \mathcal{C}} Pr_\beta(X_q = 1 | A_{\beta'}) \cdot Pr_\beta(A_{\beta'} | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= \sum_{\beta' \in \mathcal{C}} Pr_{\beta'}(X_q = 1) \cdot Pr_\beta(A_{\beta'} | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&\geq \sum_{\beta' \in \mathcal{C}} (1 - \epsilon_{prog}) Pr_\beta(A_{\beta'} | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= (1 - \epsilon_{prog}) \sum_{\beta' \in \mathcal{C}} Pr_\beta(A_{\beta'} | X_0 = x_0, \ldots, X_{q-1} = x_{q-1}) \\
&= (1 - \epsilon_{prog}).
\end{aligned}$$

Claim 4:

$$Pr_\beta(X_0 = 1) \geq 1 - \epsilon_{prog}.$$

Proof of Claim 4: The proof is similar to that for Claim 3, but simpler. Let $\mathcal{B}$ be the set of finite prefixes β' of executions $\alpha \in A_\beta$ such that β' ends at time $t_0 + \delta$. Let $\mathcal{C}$ be the set of minimal elements of $\mathcal{B}$. Note that every $\alpha \in A_\beta$ is in exactly one set of the form $A_{\beta'}$ for $\beta' \in \mathcal{C}$.
Using Claim 2, we get

$$\begin{aligned}
Pr_\beta(X_0 = 1) &= \sum_{\beta' \in \mathcal{C}} Pr_\beta(X_0 = 1 | A_{\beta'}) Pr_\beta(A_{\beta'}) \\
&= \sum_{\beta' \in \mathcal{C}} Pr_{\beta'}(X_0 = 1) Pr_\beta(A_{\beta'}) \\
&\geq \sum_{\beta' \in \mathcal{C}} (1 - \epsilon_{prog}) Pr_\beta(A_{\beta'}) \\
&= (1 - \epsilon_{prog}) \sum_{\beta' \in \mathcal{C}} Pr_\beta(A_{\beta'}) \\
&= (1 - \epsilon_{prog}).
\end{aligned}$$

We now return to the main proof. Let Y_q, $0 \leq q$, be a collection of independent 0-1 random variables such that

$$Pr(Y_q = 1) = 1 - \epsilon_{prog}.$$

By Claim 3, we have that for every $q \geq 1$, and for every $x_0, x_1, ..., x_{q-1} \in \{0, 1\}$,

$$Pr_\beta(X_q = 1 | X_0 = x_0, X_1 = x_1, \ldots, X_{q-1} = x_{q-1}) \geq Pr(Y_q = 1).$$

By Claim 4, we have that

$$Pr_\beta(X_0 = 1) \geq Pr(Y_0 = 1).$$

It follows from Lemma 2.2 that, for any $r \geq 1$,

$$Pr_\beta(\sum_{q=0}^{r-1} X_q \geq d) \geq Pr(\sum_{q=0}^{r-1} Y_q \geq d).$$

Therefore, by (19), we get

$$\begin{aligned} Pr_\beta((Dist_r = d) \cup \bar{N}) &\geq Pr(\sum_{q=0}^{r-1} Y_q \geq d) \\ &= 1 - Pr(\sum_{q=0}^{r-1} Y_q < d). \end{aligned} \tag{20}$$

Now we set $r = \lfloor \gamma_3 d + \gamma_2 \tau \rfloor$. By the definition of F^δ, we have that, for any time-unbounded execution α, if $Dist_r = d$ in α, then $\alpha \in F^\delta$. Hence, by (20), we have

$$Pr_\beta(F^\delta \cup \bar{N}) \geq 1 - Pr(\sum_{q=0}^{r-1} Y_q < d). \tag{21}$$

Now we apply Lemma 2.3, with $p = 1 - \epsilon_{prog}$, to obtain an upper bound for the probability of the sum on the right-hand side of (21):

$$Pr(\sum_{q=0}^{r-1} Y_q < d) \leq e^{-\tau}. \tag{22}$$

Then by (21) and (22), we get

$$Pr(F^\delta \cup \bar{N}) \geq 1 - e^{-\tau},$$

which completes the proof. □

We now remove the δ term.

Lemma 8.12. *Let τ be a nonnegative real number. Consider $i, j \in V$, $i \neq j$, 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 $get(m)_i$ 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

$$t_0 + (\gamma_3 d + \gamma_2 \tau) f_{prog}.$$

Then

$$Pr_\beta(F \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof. Follows since Lemma 8.11 holds for every $\delta > 0$. In detail, Lemma 8.11 says that, for every $\delta > 0$. $Pr_\beta(F^\delta \cup \bar{N}) \geq 1 - e^{-\tau}$. Note that, for $0 < \delta_1 \leq \delta_2$, we have $F^{\delta_1} \cup \bar{N} \subseteq F^{\delta_2} \cup \bar{N}$. Therefore,

$$Pr_\beta(\bigcap_{\delta > 0} F^\delta \cup \bar{N}) \geq 1 - e^{-\tau}. \tag{23}$$

We claim that

$$\bigcap_{\delta > 0} F^\delta \cup \bar{N} = F \cup \bar{N}. \tag{24}$$

One direction is obvious; we argue the other, that

$$\bigcap_{\delta > 0} F^\delta \cup \bar{N} \subseteq F \cup \bar{N}.$$

So, let $\alpha \in \bigcap_{\delta>0} F^\delta \cup \bar{N}$. If $\alpha \in \bar{N}$ then $\alpha \in F \cup \bar{N}$ and we are done. On the other hand, if $\alpha \in \bigcap_{\delta>0} F^\delta$, then for every $\delta > 0$, α contains a $get(m')_j$ event for some $m' \notin C^\beta_{\Gamma(I)}(t_0)$ at a time that is $\leq t_0 + (\gamma_3 d + \gamma_2 \tau)(f_{prog} + \delta)$. Since α cannot contain an infinite sequence of discrete events at successively

decreasing times, the only possibility is that α contains a $get(m')_j$ event for some $m' \notin C^{\beta}_{\Gamma(I)}(t_0)$ at a time that is $\leq t_0 + (\gamma_3 d + \gamma_2 \tau) f_{prog}$. Thus, $\alpha \in F$, which suffices.
Then by (23) and (24), we get that

$$Pr_\beta(F \cup \bar{N}) \geq 1 - e^{-\tau},$$

as needed. □

Next, we prove a lower bound on the probability for PC, in Lemma 8.19. In doing this, we use the equivalent WPC definition from Section 8.3.1. We decompose the analysis in terms of the number of *get* or *ack* events that have occurred so far. This requires another auxiliary definition, a version of the $WPC_{i,j}$ definition that depends on the number of *get* or *ack* events.

Definition 8.13 ($WPC_{i,j,c}(\tau)$, where $i, j \in V$, $i \neq j$, c is a positive integer, and τ is a nonnegative real)**.** *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

$$t + (\gamma_3 d + \gamma_2 \tau) f_{prog}.$$

Lemma 8.14. *Suppose $i, j \in V$, $i \neq j$, and τ is a nonnegative real. Then*

$$WPC_{i,j}(\tau) = \bigcap_{1 \leq c \leq 2nk} WPC_{i,j,c}(\tau).$$

Proof. The definitions immediate imply one direction, that

$$WPC_{i,j}(\tau) \subseteq \bigcap_{1 \leq c \leq 2nk} WPC_{i,j,c}(\tau).$$

For the other direction, that

$$\bigcap_{1 \leq c \leq 2nk} WPC_{i,j,c}(\tau) \subseteq WPC_{i,j}(\tau),$$

we use the fact that the total number of *get* and *ack* events is bounded by $2nk$: one *get* and one *ack* event for each of the n nodes for each of the $\leq k$ messages. □

For use in handling race conditions, it is also helpful to define an extension of the previous definition that includes a δ term:

Definition 8.15 ($WPC^{\delta}_{i,j,c}(\tau)$, where $i, j \in V$, $i \neq j$, c is a positive integer, and δ and τ are nonnegative reals)**.** *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

$$t + (\gamma_3 d + \gamma_2 \tau) f_{prog} + \delta.$$

Now we prove a lower bound for $WPC_{i,j,c}(\tau)$. Note that the probabilities in Lemma 8.16 are with respect to the entire probabilistic execution of $BMMB$, starting from an initial state.

Lemma 8.16. *For any $i, j \in V$, $i \neq j$, positive integer c, positive real δ and nonnegative real τ, we have:*

$$Pr(WPC^{\delta}_{i,j,c}(\tau) \cup \bar{N}) \geq 1 - e^{-\tau},$$

Proof. Fix i, j, c, δ, and τ. Define the usual notation for $P_{i,j}$ and I.
Define $\mathcal{B}^\delta$ to be the set of finite prefixes β of executions α containing at least c *get* or *ack* events, such that the c^{th} such event occurs at time t and β ends at time $t+\delta$. Let $\mathcal{C}^\delta$ be the set of minimal elements of $\mathcal{B}^\delta$. Note that every time-unbounded execution α containing at least c *get* or *ack* events is in at most one set of the form A_β for $\beta \in \mathcal{C}^\delta$.
Let $\mathcal{D}$ be the set of time-unbounded executions that contain fewer than c *get* or *ack* events. Notice that $\mathcal{D} \subseteq WPC^\delta_{i,j,c}(\tau)$

Claim 1: For every $\beta \in \mathcal{C}^\delta$,

$$Pr_\beta(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof of Claim 1: Let t be the time of the c^{th} *get* or *ack* event in β. If β contains no $get(m)_i$ event for a message $m \notin C^\beta_{\Gamma(I)}(t)$ by time t, then by definition, $A_\beta \subseteq WPC^\delta_{i,j,c}(\tau)$, so

$$Pr_\beta(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) = 1.$$

So from now on assume that β contains a $get(m)_i$ event for a message $m \notin C^\beta_{\Gamma(I)}(t)$ by time t. Fix such an m. If $m \in C^\beta_{\Gamma(I)}(t+\delta)$, then in particular, $m \in C^\beta_j(t+\delta)$, which implies that $get(m)_j$ occurs in β, so again $A_\beta \subseteq WPC^\delta_{i,j,c}(\tau)$, so

$$Pr_\beta(WPC^\delta_{i,j,m}(\tau) \cup \bar{N}) = 1.$$

So from now on assume that $m \notin C^\beta_{\Gamma(I)}(t+\delta)$.
Now we apply Lemma 8.12 to β, with $t_0 = t+\delta$, to conclude that, with probability $\geq 1-e^{-\tau}$, either there exists $m' \notin C^\beta_{\Gamma(I)}(t+\delta)$ such that $get(m')_j$ occurs by time

$$t + \delta + (\gamma_3 d + \gamma_2 \tau) f_{prog},$$

or the execution is in $\bar{N}$. For each such m', we have that $m' \notin C^\beta_{\Gamma(I)}(t)$, so we get:

$$Pr_\beta(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Then we use Claim 1 to obtain:

$$\begin{aligned}
Pr(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) &= \sum_{\beta \in \mathcal{C}^\delta} Pr_\beta(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) \cdot Pr(A_\beta) + Pr(WPC^\delta_{i,j.c}(\tau) \cup \bar{N} | \mathcal{D}) \cdot Pr(\mathcal{D}) \\
&= \sum_{\beta \in \mathcal{C}^\delta} Pr_\beta(WPC^\delta_{i,j,c}(\tau) \cup \bar{N}) \cdot Pr(A_\beta) + Pr(\mathcal{D}) \\
&\geq (1 - e^{-\tau}) \cdot Pr(\bar{\mathcal{D}}) + Pr(\mathcal{D}) \\
&\geq (1 - e^{-\tau}),
\end{aligned}$$

as needed.

□

And now we remove δ:

Lemma 8.17. *For any $i, j \in V$, $i \neq j$, positive integer c, and positive real τ, we have:*

$$Pr(WPC_{i,j,c}(\tau) \cup \bar{N}) \geq 1 - e^{-\tau}.$$

Proof. By an argument like the one used to prove Lemma 8.12. □

Lemma 8.18. *Let τ be a nonnegative real number. Then*

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

Proof. By definition of WPC and Lemma 8.14, we obtain that

$$WPC(\tau) = \bigcap_{i,j \in V, i \neq j} WPC_{i,j}(\tau) = \bigcap_{i,j \in V, i \neq j, 1 \leq c \leq 2nk} WPC_{i,j,c}(\tau).$$

Using a union bound and Lemma 8.17, we obtain:

$$Pr(WPC(\tau) \cup \bar{N}) = Pr(\bigcap_{i,j \in V, i \neq j, 1 \leq c \leq 2nk} (WPC_{i,j,c}(\tau) \cup \bar{N})) \geq 1 - 2n^3 k e^{-\tau}.$$

□

We now use Lemmas 8.18 and 8.9 to obtain our lower bound on the probability of $PC(\tau)$.

Lemma 8.19. *Let τ be a positive real number. Then*

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

Proof. Using Lemma 8.18, we obtain:

$$\begin{aligned} Pr(WPC(\tau) \cap N) &\geq Pr((WPC(\tau) \cap N) \cup \bar{N}) - Pr(\bar{N}) \\ &= Pr(WPC(\tau) \cup \bar{N}) - Pr(\bar{N}) \\ &\geq 1 - 2kn^3 e^{-\tau} - Pr(\bar{N}). \end{aligned}$$

By Lemma 8.9, $WPC(\tau) = PC(\tau)$, which completes the proof.

□

Finaly, we combine Lemma 8.10 with Lemma 8.19 and the bound for $Pr(\bar{N})$ in Lemma 4.2, and instantiate τ as $\lceil \ln(\frac{2n^3 k}{\epsilon}) \rceil$, to obtain our result for *BMMB* over the probabilistic MAC layer:

Theorem 8.20. *Let $m \in M$ and let ϵ be a real number, $0 < \epsilon < 1$. The BMMB protocol 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 $get(m)$ events, and hence, deliver events occur at all nodes in α by time

$$t_0 + \left((\gamma_3 + \gamma_2)D + ((\gamma_3 + 2\gamma_2)\lceil \ln(\frac{2n^3 k}{\epsilon}) \rceil + \gamma_3 + \gamma_2)k' \right) f_{prog} + (k' - 1) f_{ack}.$$

Proof. Let $\tau = \lceil \ln(\frac{2n^3 k}{\epsilon}) \rceil$. The theorem follows immediately from two claims:

Claim 1: Suppose $\alpha \in PC(\tau) \cap N$. Suppose an $arrive(m)_i$ event π occurs at time t_0 in α. Let k' be a positive integer such that $|K(m)| \leq k'$. Consider any node j. Then a $get(m)_j$ occurs by time

$$t_1 = t_0 + \left((\gamma_3 + \gamma_2)D + ((\gamma_3 + 2\gamma_2)\lceil \ln(\frac{2n^3 k}{\epsilon}) \rceil + \gamma_3 + \gamma_2)k' \right) f_{prog} + (k' - 1) f_{ack}.$$

Proof of Claim 1: Let $M' \subseteq M$ be the set of messages m' for which $arrive(m)_i$ precedes $clear(m')$ in α. Therefore, we have $K(m) \subseteq M'$.
Based on Lemma 8.10, and using the fact that $dist(i,j) \leq D$, by time t_1, either a $get(m)_j$ event occurs or there exists a set $M'' \subseteq M'$ with $|M''| = k'$ such that $get(m')_j$ events occur for all messages m', $m' \in M''$. In the first case, the claim holds.
So suppose that the first case does not hold and the second case does hold, that is, a $get(m)_j$ event does not occur by time t_1, but there is a set $M'' \subseteq M'$ with $|M''| = k'$ such that $get(m')_j$ events occur for all messages $m' \in M''$ by time t_1. Since $get(m)_j$ does not occur by time t_1, $clear(m)$ does not occur by time t_1. Therefore, the $arrive(m')$ events for all $m' \in M''$ precede $clear(m)$. It follows that $M'' \subseteq K(m)$. Then

because $|M''| = k'$ and $|K(m)| \leq k'$, we get $M'' = K(m)$. Since $m \in K(m)$, it follows that there is a $get(m)_j$ event by time t_1, a contradiction.

Claim 2: The probability of the event $PC(\tau) \cap N$ is at least $1 - \epsilon - nk\epsilon_{ack}$.
Proof of Claim 2: By Lemma 8.19, the probability of the event $PC(\tau) \cap N$ is at least $1 - 2n^3ke^{-\tau} - Pr(\bar{N})$. Since $\tau \geq \ln(\frac{2n^3k}{\epsilon})$, this yields that

$$Pr(PC(\tau) \cap N) \geq 1 - \epsilon - Pr(\bar{N}) \geq 1 - \epsilon - nk\epsilon_{ack}.$$

The last inequality follows from Lemma 4.2. □

8.3.4. Analysis of the Complete Algorithm
Finally, we combine our bound for the *BMMB* protocol in terms of the probabilistic abstract MAC layer (Theorem 8.20) with our results for *DMAC* to obtain a bound for the combined *BMMB*-Decay algorithm.

Theorem 8.21. *Let $m \in M$ and ϵ be a real number, $0 < \epsilon < 1$. Let $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$. The BMMB-Decay($\phi$) algorithm guarantees that, with probability at least $1-\epsilon$, the following property holds of the generated execution α:*
Suppose an $arrive(m)_i$ event π occurs in α. Let k' be a positive integer such that $|K(m)| \leq k'$.
Then $get(m)$ events, and hence, deliver 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.

Note that if k is polynomial in n, the bound reduces to $O((D + k'\Delta \log(\frac{n}{\epsilon})) \log(\Delta))$.

Proof. Choose $\epsilon_{ack} = \frac{\epsilon}{2nk}$. Theorem 8.20 implies that, with probability at least

$$1 - \frac{\epsilon}{2} - nk\epsilon_{ack} \geq 1 - \epsilon,$$

$get(m)$ events occur everywhere within time

$$\left((\gamma_3 + \gamma_2)D + ((\gamma_3 + 2\gamma_2)\lceil \ln(\frac{4n^3k}{\epsilon}) \rceil + \gamma_3 + \gamma_2)k'\right) f_{prog} + (k' - 1)f_{ack}.$$

Using the definitions of parameters for the implementation of the probabilistic layer, in Section 5.4, 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

$$O((D + \log(\frac{nk}{\epsilon})k') \log(\Delta)) + (k' - 1)O(\Delta \log(\frac{nk}{\epsilon}) \log(\Delta)) = O((D + k'\Delta \log(\frac{nk}{\epsilon})) \log(\Delta)).$$

The reason why we can use $f_{ack} = O(\Delta \log(\frac{nk}{\epsilon}) \log(\Delta))$ here is as follows. We 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

$$\epsilon_{ack} = \frac{\epsilon}{2nk\Delta} \cdot \Delta = \frac{\epsilon}{2nk}.$$

This yields, from the parameter definitions, that

$$f_{ack} = O(\Delta \log(\frac{2nk\Delta}{\epsilon}) \log(\Delta)),$$

which is

$$O(\Delta \log(\frac{nk}{\epsilon}) \log(\Delta)),$$

as needed. □

9. Conclusions

In this paper, we have shown how one can use abstract MAC layers to decompose global 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 [1, 2] and a new probabilistic layer. The basic layer is simple to use, but yields bounds that are not optimal. The probabilistic layer yields better bounds, at the cost of somewhat more difficult 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 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.

Even with the decomposition, the proofs are not trivial. Complications arise because of issues such as race conditions, the combininations of synchronous and asynchronous algorithms, and composition of probabilistic systems. Nevertheless, the analysis is not too difficult, and the results are reusable.

Some technical questions remain. For example, we wonder whether one could remove the dependence on k, the total number of messages sent in the entire execution, in the bound for multi-message broadcast (Theorem 8.21).

Other avenues for future work involve designing and analyzing other algorithms over the MAC layers, and developing and analyzing other algorithms to implement the MAC layers. For instance, as we described in the introduction, Cornejo et al. [11, 12] have developed new Neighbor Discovery algorithms over the basic abstract MAC layer, which support higher-level dynamic graph algorithms. Khabbazian et al. [14] have developed an implementation of the probabilistic abstract MAC layer based on Analog Network Coding (ANC) techniques [15], which can be combined with our high-level broadcast algorithms. Many more examples remain to be studied. We are also interested in learning how the theoretical results change in the presence of communication uncertainty, as represented by the dual graph model of Kuhn et al. [1, 2].

We hope that this work will contribute to building a comprehensive theory for wireless network algorithms, spanning all the way from the physical network level to applications.

References

[1] F. Kuhn, N. Lynch, C. Newport, The Abstract MAC layer, in: The Proceedings of the International Symposium on Distributed Computing, pp. 48–62.

[2] F. Kuhn, N. Lynch, C. Newport, The Abstract MAC layer, MIT Technical Report (MIT-CSAIL-TR-2009-021) (2009).

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

[4] R. Bar-Yehuda, O. Goldreich, 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 (1992) 104–126.

[5] D. Peleg, Time-efficient broadcasting in radio networks: A review, in: The Proceedings of The International Conference on Distributed Computing and Internet Technologies, pp. 1–18.

[6] L. Gasieniec, On efficient gossiping in radio networks, in: The Proceedings of The International Colloquium on Structural Information and Communication Complexity, pp. 2–14.

[7] A. Pelc, Algorithmic aspects of radio communication, in: The Proceedings of The International Workshop on Foundations of Mobile Computing, pp. 1–2.

[8] F. Kuhn, N. Lynch, C. Newport, The Abstract MAC layer, Distributed Computing (2011). To appear. Special issue.

[9] F. Kuhn, N. Lynch, C. Newport, The Abstract MAC Layer, Technical Report MIT-CSAIL-TR-2010-040, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 2010.

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

[11] A. Cornejo, N. Lynch, S. Viqar, J. Welch, A neighbor discovery service using an Abstract MAC layer, in: The Proceedings of Allerton Conference on Communication, Control and Computing.

[12] A. Cornejo, S. Viqar, J. Welch, Reliable neighbor discovery for mobile ad hoc networks, in: The Proceedings of Sixth ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIALM-POMC 2010), pp. 63–72.

[13] S. Dolev, S. Gilbert, M. Khabbazian, C. Newport, More channels is better: Efficient and robust wireless broadcast, 2010. Submitted for publication.

[14] M. Khabbazian, F. Kuhn, N. Lynch, M. Medard, A. ParandehGheibi, MAC design for analog network coding, 2010. Submitted for publication.

[15] S. Gollakota, D. Katabi, ZigZag decoding: Combating hidden terminals in wireless networks, in: The Proceedings of the ACM SIGCOMM Conference, volume 38, pp. 159–170.

[16] M. Adler, C. Scheideler, Efficient communication strategies for ad hoc wireless networks, Theory Comput. Syst. 33 (2000) 337–391.

[17] D. Kowalski, A. Pelc, Broadcasting in undirected ad hoc radio networks, in: The Proceedings of the International Symposium on Principles of Distributed Computing, pp. 73–82.

[18] A. Czumaj, W. Rytter, Broadcasting algorithms in radio networks with unknown topology, in: The Proceedings of the Symposium on Foundations of Computer Science, pp. 492–501.

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

[20] M. Khabbazian, D. Kowalski, F. Kuhn, N. Lynch, Decomposing broadcast algorithms using Abstract MAC layers, in: The Proceedings of Sixth ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIALM-POMC 2010), pp. 13–22.

[21] M. Khabbazian, D. Kowalski, F. Kuhn, N. Lynch, The Cost of Global Broadcast using Abstract MAC Layers, Technical Report MIT-CSAIL-TR-2010-005, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2010.

[22] S. Mitra, A Verification Framework for Hybrid Systems, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[23] D. Bertsekas, J. N. Tsitsiklis, Introduction to Probability, Athena Scientific, 2008.