MIT-CSAIL-TR-2010-005 February 9, 2010

The Cost of Global Broadcast Using Abstract MAC Layers

Majid Khabbazian, Dariusz Kowalski, Fabian Kuhn, and Nancy Lynch

The Cost of Global Broadcast Using Abstract MAC Layers

Majid Khabbazian
Massachusetts Institute of Technology
United States
majidk@mit.edu

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

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

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

February 9, 2010

Abstract

We analyze greedy algorithms for broadcasting messages throughout a multi-hop wireless network, using a slot-based model that includes message collisions without collision detection. Our algorithms are split formally into two pieces: a high-level piece for broadcast and a low-level piece for contention management. We accomplish the split using abstract versions of the MAC layer to encapsulate the contention management. We use two different abstract MAC layers: a basic non-probabilistic one, which our contention management algorithm implements with high probability, and a probabilistic one, which our contention management algorithm implements precisely.

Using this approach, we obtain the following complexity bounds: Single-message broadcast, using the basic abstract MAC layer, 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. Single-message broadcast, using the probabilistic abstract MAC layer, takes time only $O((D+\log(\frac{n}{\epsilon}))\log(\Delta))$. For multi-message broadcast, the bounds are $O((D+k'\Delta)\log(\frac{n}{\epsilon})\log(\Delta))$ 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 single message everywhere in the presence of at most k' concurrent messages.

1 Introduction

The last few years have seen an immense amount of research on algorithms for wireless ad hoc and sensor networks. Although results based on simulations have dominated the literature, there has also been a rapid growth in the amount of analytical work. Most of the theoretical work has followed one of two approaches. The first, represented, for example, by [22], analyzes wireless network algorithms using standard message-passing models, and ignores interference and other lower layer issues, assuming that they are handled by a separate Medium Access Control (MAC) layer. The second approach, represented by [3], uses models that are close to the actual physical network and requires all algorithms to handle basic communication issues.

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 simple 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.

Devising algorithms directly for the physical network, on the other hand, avoids these problems, but it requires the algorithm designer to cope with physical layer issues such as message loss due to interference and collisions. This results in complex algorithms and analyses even for very simple tasks. Studying algorithms for more complex high-level problems becomes almost prohibitively hard. Moreover, there are a variety of wireless communication models [21, 11, 20], 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.

In [16, 17], we proposed a new approach with the goal of combining the advantages of both approaches described above while avoiding their major problems. Namely, we defined an *abstract MAC layer* service that expresses the key guarantees of real MAC layers. Our 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 subdivide the effort of designing and analyzing wireless network algorithms into two manageable pieces: a piece that implements the abstract MAC layer over a physical network, and a piece that uses the abstract layer to solve higher-level problems. To illustrate our approach, we analyzed a greedy multi-message global broadcast protocol over our 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.

In [16, 17], we focused on the higher-level issues of designing and analyzing algorithms over the abstract MAC layer. However, to build a complete theory for wireless network algorithms, one must combine such results with design and analysis of lower-level algorithms that implement the abstract MAC layer over a physical network. The combination should yield realistic algorithms and analyses for high-level problems over the physical network.

Another issue that we did not address in [16, 17] is 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 completely realistic results for higher-level algorithms, one should also take such probabilities into account.

This paper: In this paper, we demonstrate that one can combine results about high-level algorithms based on an abstract MAC layer with algorithms that implement an abstract MAC layer over a physical network, and thereby obtain good overall results for high-level algorithms over a physical network. Specifically, we develop and analyze greedy algorithms for broadcasting single messages and multiple messages throughout a wireless network, using a physical network model that is slot-based and includes collisions without collision detection. Each of our algorithms is split formally into a high-level piece for broadcast and a low-level piece for contention management. We use abstract MAC layers to encapsulate the contention management.

We give a probabilistic contention management algorithm similar to those in [3, 2, 14], which uses a strategy of adjusting transmission probabilities. We use two different MAC layers: the basic non-probabilistic one from [16, 17], which our contention management algorithm implements with high probability, and a new probabilistic one, which the contention management implements exactly.

Using this approach, we obtain the following complexity bounds: Single-message broadcast, using the basic abstract MAC layer, 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. Single-message broadcast, using the probabilistic abstract MAC layer, takes time only $O((D+\log(\frac{n}{\epsilon}))\log(\Delta))$. For multi-message broadcast, the bounds are $O((D+k'\Delta)\log(\frac{nk}{\epsilon})\log(\Delta))$ 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 $O((D+k'\Delta\log(\frac{n}{\epsilon}))\log(\Delta))$. Thus, for this problem, our bounds based on the probabilistic MAC layer are significantly better than those using the basic MAC layer of [16, 17].

The single-message bounds based on the probabilistic MAC layer are the same as bounds previously obtained without such a split [3]; however, our split subdivides the analysis, thus making it clearer and simpler. The multi-message bounds are new. Indeed, our results for multi-message broadcast are sufficiently complex that they would be difficult to obtain without such a split. Thus, we believe that our approach enables the study of more difficult algorithms than would be possible otherwise. In this way, this work is an important step in developing a theory for wireless network algorithms.

Related work: The problem of broadcasting one message throughout an ad hoc radio network was first studied in [3]. That paper contains a randomized algorithm based on procedure Decay (cf. Section 5) that accomplishes the task in $O((D+\log(n/\epsilon))\log\Delta)$ steps with probability $\geq 1-\epsilon$. We obtain the same bound, showing that in this case, the analysis can be completely split into two parts without significant loss. The papers [15, 10] obtain a bound of $O((D+\log(n/\epsilon))\log(n/D))$, which represents an improvement over [3] for dense networks with large diameters. Combined, the results in [3, 10, 15] match the lower bounds of $\Omega(D\log\min\{n/D,\Delta\})$ [18] and $\Omega(\log^2 n)$ [1], for $\epsilon < 1/2$ and Δ polynomial in n.

A multi-message broadcast algorithm was given in [4]; it relies on a BFS tree built in a setup phase prior to the broadcast requests. The setup time is $O((n+D\log(n/\epsilon))\log\Delta)$, and the broadcast time is $O((k+D)\log(n/\epsilon)\log\Delta)$, both with probability $\geq 1-\epsilon$. Thus, the overall cost is $O((n+k\log(n/\epsilon)+D\log(n/\epsilon))\log\Delta)$, with probability $1-\epsilon$. Weaknesses of this approach are the need to precompute the BFS tree and the fact that all communication is relayed through a single node, the root of the tree. Our algorithm is faster than the one in [4] for $k\Delta\log(n/\epsilon) < n$. Moreover, our algorithm does not require any precomputation and is much simpler (the algorithm over the MAC layer is a trivial greedy algorithm) and more robust (the algorithm is symmetric and does not have a single point of failure). The paper [10] also contains a randomized algorithm for n simultaneous broadcasts (also known as the gossip problem) working in time $O(n\log(n/\epsilon)\log n)$ with probability $\geq 1-\epsilon$. The multi-message broadcast model in [10] differs significantly from the model used here and in [4] in that it allows intermediate nodes to combine an arbitrary amount of information into a single message, thus reducing high-level contention and leading to faster algorithms. In all of this related work, the analysis of broadcast is intertwined with that of contention management.

In addition to these results for randomized protocols, there has also been considerable work on deterministic radio network broadcast algorithms. It turns out that deterministic contention resolution is significantly harder. We mention only the strongest current bounds. For one message, it is shown in [15] that deterministic broadcast requires time $\Omega(n\log(n)/\log(n/D))$. Hence, even in networks with constant diameter, $\Omega(n)$ time is required to deterministically broadcast one message. The bound is almost matched by the best known algorithm, which allows broadcasting in time $O(n\min\{\log^2 D,\log n\})$ [10, 15]. There has also been some work on deterministic multi-message broadcast (e.g. [7, 8, 12, 6]). In particular, it was shown in [12] that all-to-all broadcast can be done in $O(n^{4/3}\log^4 n)$ time; as in [10], this assumes that an arbitrary amount of information can be packed into a single message. Deterministic gossip without combined messages was studied in [6], where an $O(n^{3/2}\,\mathrm{polylog}(n))$ algorithm was presented.

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 presents our results for single-message broadcast, and Section 7 our results for multi-message broadcast. Section 8 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. 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 we define $\Gamma(i)$ to 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

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 that is used twice later in the paper.

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

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

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)

Our results are expressed in terms of *probabilistic timed I/O automata*, as defined by Mitra [19]. PTIOAs include mechanisms (local schedulers and task schedulers) to resolve nondeterminism. 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 schedule 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 real 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 finite execution β. More precisely, we consider the conditional probability distribution on the set A_β of time-unbounded executions that extend β. We denote this probability distribution by Pr_β.

3 The Physical Model

We assume a collection of n probabilistic processes. We use probabilistic processes rather than deterministic because the MAC-layer algorithms we will consider are randomized, although the higher-level network-wide broadcast algorithms are deterministic. Using a deterministic model for MAC-layer algorithms would be problematic because of some inherent limitations of that model. For example, in the radio network model without collision detection, deterministic contention resolution (i.e., selection of a single contender) requires $\Omega(k \log \frac{n}{k})$ slots for k contenders, while non-adaptive deterministic full contention resolution solutions (i.e., each contender will transmit eventually) require $\Omega(\min(n, k^2 \log_k n))$ slots; both bounds hold even on a single-hop network [8].

We assume that time is divided into *slots*, each of real-time duration t_{slot}. 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 receiver operates at every slot, whereas the nodes decide when to transmit.

We assume that the n processes reside at the nodes of communication graph $G = (V, E)$. This is a special case of the model considered in [16, 17], with only a single, static, undirected graph G. Processes have no knowledge of the graph; in particular, they do not know their neighbors in G.

We assume a physical network, Net with collisions, and with receiver-side collision detection. When a process transmits in some slot, its message *reaches* itself and all G-neighboring processes, and no other processes. Thus, each process j, in each slot, is reached by some collection of messages (from itself and its transmitting neighbors). What j actually *receives* is defined as follows: If j is reached by its own message, then it receives 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 is happening. (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 as $\bot$. (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, $\bot$. Thus, processes cannot distinguish collisions from silence; that is, there is no collision-detection.

4 Abstract MAC Layers

4.1 The Basic MAC Layer

The *Basic Abstract MAC layer* is defined in [16, 17]. This layer provides an interface with *bcast* and *abort* inputs and *rcv* and *ack* outputs. It gives worst-case bounds for three kinds of delay, which we call *receive delay*, *acknowledgement delay*, and *progress delay*, expressed in terms of functions f_{rcv}, f_{ack}, and f_{prog}, respectively. It also has a parameter t_{abort} representing an upper bound on the possible time lag between an *abort* and a corresponding later *rcv*. For precise guarantees, we refer the reader to [16, 17].
In this paper, we use a special case of the general definitions in [16, 17]. Namely, as noted above for the Physical Network, we consider only a single, static undirected graph G. Also, since our bounds do not depend on the actual contention, but only on the maximum node degree, we use constant-valued functions for f_{rcv}, f_{ack}, and f_{prog}, and treat these as real numbers rather than functions.

4.2 The Probabilistic MAC Layer

In this section, we present a new *Probabilistic Abstract MAC Layer*, which specifies probability bounds for the three kinds of delays. The layer is implicitly parameterized by three positive real constants f_{prog}, f_{rcv}, and f_{ack} as before. In addition, it has three other parameters, ϵ_{prog}, ϵ_{rcv}, and ϵ_{ack}, representing error probabilities for attaining the three delay bounds. Finally, it has a parameter t_{abort} representing the possible time lag between an *abort* and a corresponding later *rcv*.
We model a MAC layer formally as a PTIOA Mac. To satisfy our parameterized specification, Mac must satisfy several requirements. In each case, the requirement must hold for Mac in combination with any probabilistic environment Env and the physical network Net (also modeled as PTIOAs). The composition $Mac \| Env \| Net$ is a closed probabilistic system (again a PTIOA), which yields a unique probabilistic execution.
In defining these properties, we use the following terminology. Consider a closed execution β and a *bcast* event π occurring in β. Then we say that π is *active at the end of* β provided that π is not terminated with an *ack* or *abort* in β.

1. *Non-probabilistic guarantees:* In every execution, the *Proximity*, *No duplicate receives*, *No receives after acknowledgements*, *No duplicate acknowledgements*, and *No acknowledgements after aborts* conditions from [16, 17] are satisfied.[1] Also, no *rcv* happens more than t_{abort} time after a corresponding *abort*.

2. *Probabilistic guarantees:* We have three delay bound requirements. Assume $i, j \in V$, and t is a nonnegative real.

[1]Because we are considering a single static graph only, the Proximity condition is simpler than the general one in [16, 17]. Here it says that only neighbors j of i in G can receive messages sent by process i.

(a) *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 $Pr_\beta(B|A) \geq 1 - \epsilon_{rcv}$.

(b) *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 $Pr_\beta(B|A) \geq 1 - \epsilon_{ack}$.

(c) *Progress delay 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:
  i. An $ack(m_i)_i$ for every $i \in I$,
  ii. A $rcv(m_i)_j$ for some $i \in I$, or
  iii. A rcv_j for some message whose *bcast* occurs after β.

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

5 Implementing the Abstract MAC Layer over the Physical Layer

We implement the abstract MAC layers using an algorithm *DMAC* based on a version of the Decay strategy from [3]; it is also similar to the Probability-Increase algorithm in [14]. We prove two results about *DMAC*: Theorem 5.7, which says that it implements the probabilistic abstract MAC layer (exactly), and Theorem 5.13, which says that it implements the basic abstract MAC layer with high probability.

5.1 Modified Decay

Here we describe a probabilistic contention resolution algorithm similar to the *Decay* strategy in [3]. Our algorithm differs slightly from the one in [3], in that the processes successively *increase* their transmission probabilities in a Decay phase rather than decrease them.[2] Also, in our algorithm the processes choose randomly whether to transmit in each individual slot, whereas in [3], they choose whether to drop out of an entire Decay phase. We give a lower bound on the success probability for our algorithm. The algorithm uses Δ, the maximum degree in G.

Decay:
This algorithm runs for exactly $\sigma = \lceil \log(\Delta + 1) \rceil$ slots.

[2]Thus, our strategy might be better called "Growth" rather than "Decay", but we stick with the earlier terminology.

A set I of processes, $|I| \leq \Delta$, plus another distinguished process j, participate. We assume that there is at least one process in I that participates in all slots. Other processes in I, and also j, may participate in only some slots, but once they stop participating, they do not participate in later slots.

At each slot $s = 1, \ldots, \sigma$, each participating process transmits with probability p_s, where $p_\sigma = \frac{1}{2}, p_{\sigma-1} = \frac{1}{2^2}, \ldots, p_{\sigma-s} = \frac{1}{2^{s+1}}, \ldots, p_1 = \frac{1}{2^\sigma}$.

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

From now on in the paper, we fix constants:

- $\sigma = \lceil \log(\Delta+1) \rceil$, and
- $t_{phase} = \sigma \cdot t_{slot} = (\lceil \log(\Delta+1) \rceil) t_{slot}$.

5.2 The *DMAC* Algorithm

Here we describe a MAC algorithm based on our Decay algorithm from Section 5.1. We call our algorithm *DMAC*(ϕ), where ϕ is a positive integer parameter indicating the number of Decay phases that are executed.

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

Each MAC layer process i that receives a message from its environment, via a $bcast(m)_i$ event, starts executing Decay with message m (and a unique message identifier) at the beginning of the next Decay phase. Process i executes exactly ϕ Decay phases, and then outputs $ack(m)_i$ at the end of the final phase. However, if process i receives an $abort(m)_i$ from the environment before it performs $ack(m)_i$, it performs no further transmission on behalf of message m and does not perform $ack(m)_i$.

Meanwhile, process i tries to receive, in every slot. When it receives any message m' from a neighbor (not from itself) for the first time on the physical network, it delivers that to its environment with a $rcv(m')_i$ event, at a real time 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 $DMAC$

We prove five lemmas giving properties of $DMAC(\phi)$. Throughout this subsection, we let ϕ be a positive integer, Env be any probabilistic environment, and consider the unique probabilistic execution of $DMAC(\phi)\|Env\|Net$. We begin with a result that holds for every time-unbounded execution, without any mention of probabilities.

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 t_{slot} time 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 time is at most t_{slot}. □

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 the Lemma 5.1 to obtain a version of the progress delay bound.

Lemma 5.4. *Let $j \in V$, and let g and h be positive integers. Let β be a closed execution that ends at time t, where $(g-1)t_{phase} \leq t < g \cdot t_{phase}$. Let I be the set of neighbors of j that have active bcasts at the end of β, where $bcast(m_i)_i$ is the bcast at i. Suppose that I is nonempty. Suppose that no $rcv(m_i)_j$ occurs in β, for any $i \in I$.*
Define the following sets of time-unbounded executions that extend β:

- *A, the executions in which no $abort(m_i)_i$ occurs for any $i \in I$.*
- *B, the executions in which, by the end of Decay phase $g+h$, at least one of the following occurs:*
  1. *A $rcv(m_i)_j$ for some $i \in I$, or*
  2. *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 $\epsilon > 0$ be a positive real. Suppose that ϕ, the parameter for $DMAC(\phi)$, 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 a $bcast(m)_i$ at time t, where $(g-1)t_{phase} \leq t < gt_{phase}$. 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 $q = 1, \dots, \phi$, 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 $x_1, x_2, ..., x_{q-1} \in \{0, 1\}$,

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

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 $Pr_\beta(\bar{A} \cup B) \geq Pr_\beta(B|A)$. □

Lemma 5.6. *Let $\epsilon > 0$ be a positive real. Suppose that $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon}) \rceil$. Let $i \in V$. Let β be any closed prefix of a time-unbounded execution that ends with a $bcast(m)_i$ at time t, where $(g-1)t_{phase} \leq t < g \cdot t_{phase}$. 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, as functions of ϵ, h, and ϕ (Δ, t_{phase}, and t_{slot}, have all been fixed).

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

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 $f_{rcv} = (\phi + 1)t_{phase}$ and $\epsilon_{rcv} = \epsilon$. Suppose that $Pr_\beta(A) > 0$. Choose g so that $(g-1)t_{phase} \leq t < g \cdot t_{phase}$. 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 $(g + \phi)t_{phase}$ and $t \geq (g-1)t_{phase}$, we have that Decay phase $g + \phi$ ends by time $\leq t + (\phi + 1)t_{phase}$. 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 $f_{ack} = (\phi + 1)t_{phase}$ and $\epsilon_{ack} = \epsilon\Delta$.
Finally, Lemma 5.4, Conclusion 1, implies that the probabilistic execution satisfies the progress delay bound. Here, we use $f_{prog} = (h + 1)t_{phase}$ 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 only 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 several constants:

- ϵ, a real number, where $0 < \epsilon \leq 1$.
- 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 individual properties.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1}) \rceil$. This is the number of Decay phases we will consider for the acknowledgement bound.
- $h = \log_{8/7}(1/\epsilon_1)$, the real number such that $(\frac{7}{8})^h = \epsilon_1$.

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

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

- $t_{abort} = t_{slot}$.

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 $f_{ack} = (\phi+1)\cdot t_{phase}$. 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., [5]) 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 probabilistic execution yields an execution that 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 Single-Message Broadcast Algorithms

In this section, we present a single-message global broadcast algorithm that uses either a basic or probabilistic abstract MAC layer. We analyze the algorithm in both cases, combining bounds in Section 5 for our MAC layer implementation with higher-level analysis of the global broadcast algorithm. The bounds, in Theorems 6.3 and 6.11, 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.
For the single-message broadcast problem, we assume that a single message begins at some distinguished node i_0.

6.1 Basic Definitions and Results

Definition 6.1 (*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.*

Lemma 6.2. *Consider a 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 that submits at most b bcasts in any execution and never submits an abort, and Net is our physical layer. Then in this system:*

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

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. □

6.2 Algorithm Using Decay

Our broadcast algorithm is a simple flooding algorithm. It is essentially the *Basic Multi-Message Broadcast* algorithm of [16, 17], specialized to one message, and combined with our Decay implementation of the MAC layer. The combination is similar to the global broadcast algorithm in [3]. We parameterize the algorithm with the number ϕ of Decay phases.

> *BSMB-Decay*(ϕ):
> We define the Basic Single-Message Broadcast ($BSMB$) algorithm to be the same as the Basic Multi-Message Broadcast ($BMMB$) algorithm from [16, 17], but specialized to the case of a single message, and modified so that the message starts in the state of a designated initial node i_0, rather than arriving from an external environment. *BSMB-Decay*(ϕ) consists of the $BSMB$ algorithm composed with $DMAC(\phi)$, our implementation of the abstract MAC layers.

In the following subsections, we analyze this algorithm, using abstract MAC layers. In Section 6.3, we consider the basic abstract MAC layer, and in Section 6.4, the probabilistic abstract MAC layer. We use different values of ϕ in these two subsections.
From now on in this section, when we talk about "executions", we mean executions of *BSMB-Decay*(ϕ) together with our physical network and environment.

6.3 Analysis Using Basic Abstract MAC

In this section, we fix 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 node.
- $a = n(\Delta + 2)$. This is a bound on the total number of external MAC layer events.
- $\epsilon_1 = \frac{\epsilon}{2a}$.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1}) \rceil$.

Using the basic abstract MAC layer, with precise bounds from Section 5.5, we obtain an upper bound for the time to deliver the message everywhere with probability at least $1 - \epsilon$.

Theorem 6.3. *The BSMB-Decay(ϕ) algorithm guarantees that, with probability at least* $1 - \epsilon$, *rcv events occur at all nodes* $\neq i_0$ *by time*

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

Proof. Theorem 3.2 of [16, 17] implies that 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 $t_{phase} = (\lceil \log(\Delta + 1) \rceil) t_{slot}$, 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

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

Thus, 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 bound.
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, Theorem 5.13 implies that the MAC layer achieves the progress bound f_{prog} with probability at least $1-\epsilon$. That in turn means that the entire system achieves the required message delivery bound with probability at least $1-\epsilon$. □

6.4 Analysis Using Probabilistic Abstract MAC

In this section, we prove another upper bound on the message delivery time for the $BSMB$-$Decay(\phi)$ protocol, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil$. This time, we use our probabilistic MAC layer. Now we improve the upper bound of the previous section to $O((D+\log(\frac{n}{\epsilon}))\log(\Delta))$. This yields the same bound as in [3], 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 these parameters. Then, in Section 7.4.4, we replace the abstract layer with a Decay-based implementation and combine bounds for that implementation with the bounds for $BSMB$ to obtain our final result, Theorem 6.11.
In Sections 6.4.1-6.4.2, 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 6.4.3, we consider the system $BSMB$-$Decay(\phi)\|Env\|Net$, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil$, and Env is some probabilistic environment.

6.4.1 Basic Definitions

We define

$$\gamma_1 = \frac{3}{1-\epsilon_{prog}} \quad \text{and} \quad \gamma_2 = \frac{2}{1-\epsilon_{prog}}. \tag{5}$$

Definition 6.4 (Progress condition $PC_j(\tau)$, where $j \in V - \{i_0\}$ and τ is a nonnegative real)**.** *We say that $\alpha \in PC_j(\tau)$ if a get_j event occurs in α by time*

$$(\gamma_1 dist(i_0, j) + \gamma_2 \tau) f_{prog}.$$

We define the set of executions $PC(\tau)$ as

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

Because of some issues involving race conditions, it is useful to define a generalization of the PC definition that allows a small amount of slack:

Definition 6.5 ($PC_j^{\delta}(\tau)$, where $j \in V - \{i_0\}$ and δ and τ are nonnegative reals)**.** *We say that $\alpha \in PC_j^{\delta}(\tau)$ if a get_j event occurs in α by time*

$$(\gamma_1 dist(i_0, j) + \gamma_2 \tau)(f_{prog} + \delta).$$

Also, we define:

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

6.4.2 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 real work is done in the following lemma, which bounds the probability of $PC_j^{\delta}(\tau)$.

Lemma 6.6. *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. 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 get_{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

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

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 get_{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 $r = \lfloor \gamma_1 d + \gamma_2 \tau \rfloor$. 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. □

Lemma 6.7. *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 6.6 holds for every $\delta > 0$. In a bit more detail, Lemma 6.6 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 get_j event at a time that is $\leq (\gamma_1 d + \gamma_2 \tau)(f_{prog} + \delta)$. 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 get_j event at a time that is $\leq (\gamma_1 d + \gamma_2 \tau) f_{prog}$. 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 6.8. *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 6.7, 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 6.9. *Let τ be a positive real number. Then*

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

Proof. Using Lemma 6.8, 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 the previous result with an upper bound on the probability of $\bar{N}$.

Theorem 6.10. *Let ϵ be a real number, $0 < \epsilon \leq 1$. The BSMB protocol guarantees that, with probability at least*

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

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

$$(\gamma_1 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}.$$

Proof. By Lemmas 6.9 and 6.2, with probability at least

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

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

$$(\gamma_1 D + \gamma_2 \tau)f_{prog}.$$

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

6.4.3 Analysis of the Complete Algorithm

Now we combine the bound for the *BSMB* protocol in terms of the probabilistic abstract MAC layer with our bound for *DMAC* to obtain a bound for the combined *BSMB-Decay* algorithm.

Theorem 6.11. *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 $1 - \epsilon$, get 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 6.10, applied with ϵ in that theorem instantiated as our $\frac{\epsilon}{2}$, implies that, with probability at least

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

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

$$(\gamma_1 D + \gamma_2 \ln(\frac{n}{\epsilon}))f_{prog}.$$

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

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

□

7 Multi-Message Broadcast Algorithms

Now we present a multi-message global broadcast algorithm that uses either a basic or probabilistic abstract MAC layer. We prove 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; these appear in Theorems 7.4 and 7.21, Our results assume a bound k on the number of messages that arrive from the environment during the entire execution.

7.1 Algorithm Using Decay

We consider the Basic Multi-Message Broadcast algorithm combined with a Decay implementation. We parameterize the combined algorithm with the number ϕ of Decay phases.

> *BMMB-Decay*(ϕ):
> The algorithm consists of the Basic Multi-Message Broadcast (BMMB) algorithm from [16, 17], composed with $DMAC(\phi)$.

In the following subsections, we analyze this algorithm, using abstract MAC layers. In Section 7.3, we consider the basic abstract MAC layer, and in Section 7.4, the probabilistic abstract MAC layer. We use different values of ϕ in these two subsections.
From now on in this section, when we talk about "executions", we mean executions of *BMMB-Decay*(ϕ) together with our physical network and an environment.

7.2 Basic definitions

Definition 7.1 (*Clear* events)**.** *Let α be an execution in N and $m \in M$ be a message for which an arrive(m) event occurs in α. The event clear(m) is defined to be the final ack(m) event in α.*

Definition 7.2 (The Set $K(m)$)**.** *Let α be an execution in N and $m \in M$ 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.*

7.3 Analysis Using Basic Abstract MAC

In this section, we fix 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.
- $\epsilon_1 = \frac{\epsilon}{2a}$.
- $\phi = \lceil 8\Delta \ln(\frac{\Delta}{\epsilon_1}) \rceil$.

Using the basic abstract MAC layer, with precise bounds from Theorem 5.13, we obtain an upper bound for the time to deliver any particular message everywhere with probability at least $1 - \epsilon$.

Definition 7.3 (*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.*

Theorem 7.4. *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 occur at all processes in α within time

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

of the time of the arrive(m) event.

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 [17] implies that 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 $1 - \epsilon$, whenver 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 the MAC layer achieves the progress bound f_{prog} and the acknowledgment bound f_{ack} with probability at least $1-\epsilon$. That in turn means that the entire system achieves the required message delivery bounds with probability at least $1 - \epsilon$. □

7.4 Analysis Using Probabilistic Abstract MAC

In this section, we prove another upper bound on the message delivery time for the *BMMB-Decay*(ϕ) protocol, where $\phi = \lceil 8\Delta \ln(\frac{1}{\epsilon})\rceil$. This time, we use our probabilistic MAC layer. Now we improve the upper bound of the previous section to $O((D + k'\Delta \log(\frac{nk}{\epsilon})) \log(\Delta))$.
In our analysis, we first assume a probabilistic abstract MAC layer with parameters f_{prog}, f_{rcv}, f_{ack}, ϵ_{prog}, ϵ_{rcv}, and ϵ_{ack} and analyze the complexity of $BMMB$ in terms of these parameters. Then, in Section 7.4.4, we replace the abstract layer with a Decay-based implementation and combine bounds for that implementation with the bounds for $BMMB$ to obtain our final result, Theorem 7.21.
We divide up our analysis of $BMMB$ over the abstract layer in an interesting way. First, in Section 7.4.1, we define a new progress condition PC. Then, in Section 7.4.2, we prove a non-probabilistic bound on the message delivery time in executions that are "well-behaved", in the sense that they satisfy this progress condition, and also are "nice" as defined in Section 6.4. Finally, in Section 7.4.3, we bound the probability that an execution is well-behaved and use this to infer our overall probabilistic bound on message delivery time.
In Sections 7.4.1-7.4.2, 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 7.4.3, 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.

7.4.1 Well-Behaved Executions

We begin with basic definitions. The first identifies the messages whose processing is completed at a particular node by a designated time:

Definition 7.5. *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 $i \in I$.

The second defines PC.

Definition 7.6 (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 $m \notin C_{\Gamma(I)}^{\alpha}(t)$ occurs in α by time t, then a $get(m')_j$ event for some message $m' \notin C_{\Gamma(I)}^{\alpha}(t)$ occurs by time

$$t + (\gamma_1 d + \gamma_2 \tau) f_{prog}.$$

We define the set of executions $PC(\tau)$ as

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

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 intervals that begin with *get* and *ack* events, rather than intervals that begin at arbitrary 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 7.4.3.

Definition 7.7 (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

$$t + (\gamma_1 d + \gamma_2 \tau) f_{prog}.$$

We define the set of executions $WPC(\tau)$ as

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

Lemma 7.8. *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 $t + (\gamma_1 d + \gamma_2 \tau) f_{prog}$. 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 $t + (\gamma_1 d + \gamma_2 \tau) f_{prog}$. 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 $t' + (\gamma_1 d + \gamma_2 \tau) f_{prog} \leq t + (\gamma_1 d + \gamma_2 \tau) f_{prog}$. It follows that $\alpha \in PC_{i,j}(\tau)$, as needed.

□

The following lemma follows immediately from Lemma 7.8.

Lemma 7.9. *For every nonnegative real τ:*

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

7.4.2 Message Delivery Guarantee for Well-Behaved Executions

In this subsection, we prove a non-probabilistic upper bound on message delivery time in well-behaved executions. In addition to assuming that executions are in $PC(\tau)$, we also assume that they are "nice", that is, in N as defined in Section 7.2.

Lemma 7.10. *Let τ be a positive integer. For integers $l \geq 1$, define*

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

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

  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 write $P_{i,j}$ as $i = i_0, i_1, \ldots, i_d = j$ and let $I = \{i_1, \ldots, i_d\}$. If $m \in C_j^\alpha(t_0)$ 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

  $$t_0 + (\gamma_1 d + \gamma_2 \tau) f_{prog} < t_{d,1}.$$

  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

$$t_{e'+\tau+1,l-1} + f_{ack} + (\gamma_1\tau + \gamma_2\tau)f_{prog} \leq t_{e'+2\tau+1,l-1} + f_{ack}.$$

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

$$t_{d-r+\tau+1,l-1} + f_{ack} + (\gamma_1 r + \gamma_2\tau)f_{prog} \leq t_{d,l}.$$

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

$$t_{d+1,l-1} + f_{ack} + (\gamma_1 r + \gamma_2\tau)f_{prog} \leq t_{d,l}.$$

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.

□

7.4.3 Probabilistic Upper Bound on Message Delivery Time

In this section, we prove a lower bound on the probability of the event $N \cap PC(\tau)$. We then tie all the results together in Theorem 7.20.

We first prove a lemma bounding the probability of short message propagation time between two 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 the sense defined in Section 7.2.

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 used in the lemma statement is defined in Section 2.3.

Lemma 7.11. *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

$$t_0 + (\gamma_1 d + \gamma_2 \tau)(f_{prog} + \delta).$$

Then

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

Proof. 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 $bcast(m')_{i_e}$ 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 $r = \lfloor \gamma_1 d + \gamma_2 \tau \rfloor$. 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. $\square$

Lemma 7.12. *Let τ be a nonnegative real number. Consider two processes i and j, write $P_{i,j}$ as $i = i_0, i_1, i_2, \ldots, i_d = j$, and let $I = \{i_1, \ldots, i_d\}$.*
Let β be a finite execution of the BMMB protocol that ends at time t_0. Assume that there exists $m \notin C^\beta_{\Gamma(I)}(t_0)$ such that a $bcast(m)_i$ event occurs in β.
Let F be the subset of A_β in which there exists $m' \notin C^\beta_{\Gamma(I)}(t_0)$ for which a $get(m')_j$ event occurs by time

$$t_0 + (\gamma_1 d + \gamma_2 \tau) f_{prog}.$$

Then

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

Proof. Follows since Lemma 7.11 holds for every $\delta > 0$. In a bit more detail, Lemma 7.11 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 $\leq t_0 + (\gamma_1 d + \gamma_2 \tau)(f_{prog} + \delta)$. Since α cannot contain an infinite sequence of discrete events at successively decreasing times (a basic property of timed executions for TIOA), 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 $\leq t_0 + (\gamma_1 d + \gamma_2 \tau) f_{prog}$. 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. □

We use Lemma 7.12 to prove a lower bound on the probability for PC, in Lemma 7.19. In doing this, we use the equivalent WPC definition. We decompose the analysis in terms of the number of *get* or *ack* events that have occurred so far. This requires an auxiliary definition, a version of the $WPC_{i,j}$ definition that depends on the number of *get* or *ack* events.

Definition 7.13 ($WPC_{i,j,c}(\tau)$, 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

$$t + (\gamma_1 d + \gamma_2 \tau) f_{prog}.$$

Lemma 7.14. *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 processes for each of the $\leq k$ messages. $\square$

For use in handling race conditions, it will also be helpful to define a slight extension of the previous definition:

Definition 7.15 ($WPC^{\delta}_{i,j,c}(\tau)$, where $i, j \in V$, $i \neq j$, c is a positive integer, 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

$$t + (\gamma_1 d + \gamma_2 \tau) f_{prog} + \delta.$$

Now we can prove a lower bound for $WPC_{i,j,c}(\tau)$. Note that the probabilities in Lemma 7.16 are with respect to the entire probabilistic execution of $BMMB$, starting from an initial state.

Lemma 7.16. *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 7.12 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

$$t + \delta + (\gamma_1 d + \gamma_2 \tau) f_{prog},$$

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.

□

Lemma 7.17. *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 7.12. □

Lemma 7.18. *Let τ be a nonnegative real number. Then*

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

Proof. By definition of WPC and Lemma 7.14, 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 7.17, we obtain:

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

□

We now use Lemmas 7.18 and 7.9 to obtain a lower bound on the probability of $PC(\tau)$.

Lemma 7.19. *Let τ be a positive real number. Then*

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

Proof. Using Lemma 7.18, we obtain:

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

By Lemma 7.9, $WPC(\tau) = PC(\tau)$, which completes the proof.

□

We combine Lemma 7.10 with Lemma 7.19 and the bound for $Pr(\bar{N})$ in Lemma 6.2, and instantiate τ as $\lceil \ln(\frac{2n^3k}{\epsilon}) \rceil$, to obtain our result for $BMMB$ over the probabilistic MAC layer:

Theorem 7.20. *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 occur at all nodes in α by time

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

Proof. Let $\tau = \lceil \ln(\frac{2n^3k}{\epsilon}) \rceil$. 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 process j. Then a $get(m)_j$ occurs by time

$$t_1 = t_0 + \left((\gamma_1 + \gamma_2)D + ((\gamma_1 + 2\gamma_2)\lceil \ln(\frac{2n^3k}{\epsilon}) \rceil + \gamma_1 + \gamma_2)k' \right) f_{prog} + (k' - 1)f_{ack}.$$

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 7.10, 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 7.19, 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 6.2. □

7.4.4 Analysis of the Complete Algorithm

Finally, we combine the bound for the $BMMB$ protocol in terms of the probabilistic abstract MAC layer with the bound for $DMAC$ to obtain a bound for the combined $BMMB$-$Decay$ algorithm.

Theorem 7.21. *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 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 7.20 implies that, with probability at least

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

get(m) events occur everywhere within time

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

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. $\square$

8 Conclusions

It remains to determine whether it is possible to remove the dependence on k, the total number of messages ever sent, in the bound for multi-message broadcast in Theorem 7.21.
In related work with Viqar and Welch [9], we are developing Neighbor Discovery algorithms over the basic MAC layer of [16, 17]. This enables the construction of high-level dynamic graph models like those used in [22] over an abstract MAC layer, and so supports the analysis of pre-existing algorithms based on dynamic graphs, in terms of abstract MAC layers and thus in terms of physical network models. We are also studying alternative implementations of abstract MAC layers, based on Zig-Zag Decoding techniques [13].
It remains to study many more algorithms over abstract MAC layers, including algorithms for basic tasks like Neighbor Discovery and Unicast with Acknowledgement, and algorithms for more complex communication, data-management and coordination tasks. It also remains to consider alternative techniques for implementing abstract MAC layers, including algorithms that try to avoid contention and those that use coding techniques to make progress in the face of contention. Finally, we are interested in learning how the theoretical results change in the face of communication uncertainty.

References

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing*, 5:67–71, 1991.

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

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

[5] D.P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability.* Athena Scientific, 2008.

[6] M. Christersson, L. Gasieniec, and A. Lingas. Gossiping with bounded size messages in ad hoc radio networks. In *The Proceedings of the International Colloquium on Automata, Languages and Programming*, pages 377–389, 2002.

[7] M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.

[8] A.E.F. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *The Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 709–718, 2001.

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

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

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

[12] L. Gasieniec, T. Radzik, and Q. Xin. Faster deterministic gossiping in directed ad hoc radio networks. In *The Proceedings of the Scandinavian Workshop on Algorithm Theory*, pages 397–407, 2004.

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

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

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

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

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

[18] E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

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

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

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

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