Multi-Message Broadcast with Abstract MAC Layers and Unreliable Links[*]

Mohsen Ghaffari
MIT
ghaffari@csail.mit.edu

Erez Kantor
MIT
erezk@csail.mit.edu

Nancy Lynch
MIT
lynch@csail.mit.edu

Calvin Newport
Georgetown University
cnewport@cs.georgetown.edu

Abstract

We study the multi-message broadcast problem using abstract MAC layer models of wireless networks. These models capture the key guarantees of existing MAC layers while abstracting away low-level details such as signal propagation and contention. We begin by studying upper and lower bounds for this problem in a *standard abstract MAC layer model*—identifying an interesting dependence between the structure of unreliable links and achievable time complexity. In more detail, given a restriction that devices connected directly by an unreliable link are not too far from each other in the reliable link topology, we can (almost) match the efficiency of the reliable case. For the related restriction, however, that two devices connected by an unreliable link are not too far from each other in geographic distance, we prove a new lower bound that shows that this efficiency is impossible. We then investigate how much extra power must be added to the model to enable a new order of magnitude of efficiency. In more detail, we consider an *enhanced abstract MAC layer model* and present a new multi-message broadcast algorithm that (under certain natural assumptions) solves the problem in this model faster than any known solutions in an abstract MAC layer setting.

1 Introduction

Most existing work on distributed algorithms for wireless networks assumes low-level synchronous models that require algorithms to deal directly with link-layer issues such as signal fading (e.g., [22, 20, 13]) and contention (e.g., [2, 16, 28, 12]). These low-level models are appropriate for answering basic science questions about the capabilities and limits of wireless communication. We argue, however, that they are often *not* appropriate for designing and analyzing algorithms meant for real world deployment, because: (1) they fail to capture the unpredictable reality of real radio signal propagation (which tends not to follow simple collision or fading rules [33]); (2) they do not address issues like network co-existence (it is rarely true that your algorithm is alone in using the wireless channel); and (3) they ignore the presence of general purpose MAC layers which are standard and hard to bypass in existing devices.

[*] Supported in a part by AFOSR FA9550-13-1-0042 and NSF grants Nos. CCF-0939370, CCF-1217506, and CCF-AF-0937274.

In [29, 30], we introduced the *abstract MAC layer* approach as an alternative to low-level models for studying wireless algorithms. This approach moves the algorithm designer up the network stack by modeling the basic guarantees of most existing wireless MAC layers. In doing so, it abstracts away low level issues such as signal fading and contention, instead capturing the impact of this behavior on higher layers with model uncertainty. Because abstract MAC layers are defined to maintain the basic guarantees of most standard wireless MAC layers, algorithms developed in such models can be deployed on existing devices while maintaining their formally proved properties.

In this paper, we study the basic communication primitive of *multi-message broadcast* (a subset of devices start with one or more messages they need to disseminate to the whole network) in abstract MAC layer models that include unreliable links. We produce new upper and lower bounds, and explore new model variants. Our results, summarized below and in Figure 1, provide new theoretical insight into the relationship between unreliability and efficiency, and identify practical algorithmic strategies.

Abstract MAC Layer Models. Abstract MAC layer models provide devices with an acknowledged local broadcast primitive that guarantees to deliver a message to a device's reliable neighbors (captured by a graph G) and possibly some arbitrary subset of additional unreliable neighbors (captured by a graph G'). At some point after the message is delivered, the sender receives an acknowledgment.[1] The performance of the model in a given execution is defined by two constants: F_{ack}, the maximum time for a given local broadcast to complete and be acknowledged, and F_{prog}, the maximum time for a device to receive *some* message when at least one nearby device is broadcasting. We note that in both theory and practice, $F_{prog} \ll F_{ack}$.[2]

Results. In this paper, we consider the multi-message broadcast (MMB) problem. This problem distributes $k \geq 1$ messages to devices at the beginning of an execution, where k is not known in advance. It is considered solved once all messages are delivered to all nodes. We begin by studying a natural MMB strategy called Basic Multi-Message Broadcast (BMMB) in what we call the *standard abstract MAC layer* model: a basic model that captures the key guarantees of existing MAC layers. The BMMB algorithm implements an expected strategy for broadcast: on first receiving a message m, from the environment or from another device, add it to your FIFO queue of messages to broadcast; going forward, discard any additional copies of m that you receive. In previous work [30], we proved that BMMB solves the problem in $O(DF_{prog} + kF_{ack})$ time in the standard abstract MAC layer model under the strong assumption that $G' = G$, i.e., there are no unreliable links, and D is the diameter of G. In the first part of this paper, we investigate the performance of strategy in the presence of unreliability.

We begin by considering the case where G' is arbitrary; i.e., there are no constraints on the structure of unreliable links. Under this pessimistic regime, we reanalyze BMMB, proving a new

[1] The acknowledgment in this model describes the behavior of a standard MAC layer asking for the next message to broadcast from the send queue; i.e., after a CSMA back-off protocol finishes with a given packet. It does not represent an acknowledgment explicitly sent from the receivers.

[2] From a theory perspective, we note that standard probabilistic strategies like *decay* (cycle through an exponentially decreasing series of broadcast probabilities), when analyzed in graph-based, low-level wireless models, offer F_{prog} values that are polylogarthmic in the maximum possible contention, while F_{ack} values can be linear (or worse) in this same parameter (see [18] for more discussion). From a practical perspective, this gap is easily demonstrated. Consider, for example, a star network topology centered on device u where all points in the star have a message to broadcast. If these nodes are using standard back-off style strategies, u will receive *some* message quickly. But regardless of how contention is managed, there are some points in the star that will have to wait a long time (i.e., linear in the star size) for their turn.

Model/G' Const.	$G' = G$	r-Restricted	Grey Zone
Standard	$O(DF_{prog} + kF_{ack})$ [30]	$O(DF_{prog} + rkF_{ack})$	$\Theta((D+k)F_{ack})$
Enhanced	*same as grey zone*	*open*	$O((D + k\log n + \log^3 n)F_{prog})$

Figure 1: A summary of results categorized by model type and constraints assumed regarding G'. With the exception of the $G' = G$ result for the standard model, the results in this table are proved for the first time in this paper. We note that the grey zone result for the standard model summarizes two separate results: an upper bound and matching lower bound. These two results also hold for arbitrary G'.

guarantee of $O((D+k)F_{ack})$ time, which is (potentially) much slower than what is possible when $G' = G$. The presence of unreliable edges, it turns out, breaks the careful induction at the core of the $G' = G$ result, as they allow old messages to arrive unexpectedly from farther away in the network at inopportune points in an execution.

Not satisfied with this slow-down, we then consider the case of an *r-restricted G'*—a natural constraint that only allows G' edges between nodes within r hops in G. Under these constraints, we can now show that BMMB solves the problem in $O(DF_{prog} + r \cdot k \cdot F_{ack})$ time, which is close to the $G' = G$ case for small r. This proof discards the core strategy of the $G' = G$ case, which depends heavily on the lack of unreliable links, and instead uses a new type of pipelining argument that carefully accounts for the possible message behavior over r-restricted G' links.

We conclude our investigation of BMMB by studying the *grey zone* constraint [4, 19]: a natural geographic restriction on G' that generalizes the unit disk graph model. Here we prove the perhaps surprising lower bound result that *every MMB algorithm* requires $\Omega((D+k)F_{ack})$ time, in the worst case, to solve the problem under this constraint. This result establishes the optimality of our analysis of BMMB under the grey zone constraint, as well as for arbitrary G', and opens an intriguing gap between the grey zone and r-restricted assumptions. At the core of this lower bound is a careful scheduling strategy that synchronizes broadcasters in two parallel lines to a sufficient degree to allow their messages to cause mutual delay.

Having established the limits of MMB in the standard abstract MAC layer model, we then explore the possibility of adding more power to the model to enable more efficient solutions. In particular, we use the enhanced abstract MAC layer model of [30] which also allows nodes to abort transmissions in progress and use knowledge of F_{prog} and F_{ack}. Combining this model with the grey zone constraint on G', we describe and analyze a new algorithm, which we call Fast Multi-Message Broadcast (FMMB), that solves the MMB problem in $O((D\log n + k\log n + \log^3 n)F_{prog})$ time (with high probability in the network size, n[3])—avoiding an F_{ack} term altogether. This algorithm begins by building a maximal independent set (a subroutine of independent interest), then efficiently gathers and spreads messages over the overlay network defined by these nodes. We note that the assumptions that separate the enhanced from standard model were chosen in part because they are feasible to implement using existing technology.

Discussion. From a theoretical perspective, the new upper and lower bounds proved in this paper emphasize the perhaps surprising insight that the efficiency of message dissemination depends on the *structure* of unreliability, not the *quantity*. We are able, for example, to solve MMB fast with an r-restricted G'. This constraint allows for a large number of unreliable edges in every neighborhood, and only forbids these edges from covering long distances in G. Our lower bound, on the other

[3] We define high probability to be at least $1 - 1/n$. We note that BMMB's guarantees are deterministic but that our lower bound works even for randomized solutions.

hand, demonstrates that even a small number of unreliable edges is sufficient to slow down any MMB solution, so long as these edges are allowed to cover large distances in G.

From a practical perspective, our efficient time bounds for BMMB under the (reasonable) r-restricted assumption helps explain why straightforward flooding strategies tend to work well in real networks. In addition, our enhanced MAC layer results provide feedback to MAC layer designers, indicating that the ability to abort messages might prove crucial for enabling more efficient distributed protocols running on these layers.

Related Work. The study of broadcast in wireless networks has a long line of history, dating back to 1985 work of Chalamatac and Kutten[5], and has since received a vast amount of attention (see e.g., [3, 1, 26, 11, 31, 27, 15, 23, 17, 34, 14]). Most of this existing work deals directly with low-level issues such as managing contention on the shared medium.

The *abstract MAC layer* model was proposed in [29, 30] as an alternative approach, which moves up the network stack and abstracts away low level issues with model uncertainty. This model has since been used to study a variety of problems; e.g., [24, 9, 10, 25, 7, 6, 32]. Most relevant to this paper is the work of [29] and subsequently [24], which study broadcast in various abstract MAC layer models, but under the assumption of no unreliable edges.

A core property of the abstract MAC layer models studied in this paper, by contrast, is the presence of unreliable links in addition to reliable links. A lower level model also assuming these dual link types was introduced by Clementi et al. [8], where it was called the *dynamic fault* model. We independently reintroduced the model in [29] with the name *dual graph* model. By now it is well-known that most problems that are simple in the absence of unreliability (when $G' = G$), become significantly harder in its presence (when $G' \neq G$); e.g., [29, 18, 19, 4]. For instance, in the dual graph model with an offline adaptive adversary, single-message broadcast require $\Omega(n)$ rounds, even in constant diameter graphs [29]. We emphasize, however, that this existing work on dual graphs focuses on low level models, whereas this paper carries this behavior to a higher level of abstraction.

2 Model and Problem

There is no single abstract MAC layer model, but instead many different such models that all share the same strategy of abstracting standard wireless link layer behavior and therefore preventing the algorithm from having to deal directly with low level wireless behavior. Below we define the basics shared by these models, then specify the two variants studied in this paper. We conclude by formalizing the multi-message broadcast problem.

Abstract MAC Layer Basics. Abstract MAC layer models typically define the connectivity of a radio network with a pair of graphs, G and G', where $G = (V, E)$, $G' = (V, E')$, and $E \subseteq E'$. The n vertices in V correspond to the wireless devices (which we call *nodes* in this paper), while the edges in G and G' describe the communication topology. At a high-level, edges in E indicate reliable links over which the model can always deliver messages, while edges in $E' \setminus E$ indicate unreliable links over which the model sometimes delivers messages and sometimes does not.

The model provides an acknowledged local broadcast primitive. To simplify the definition of this primitive, assume without loss of generality that all local broadcast messages are unique. When a node $u \in V$ *broadcasts* a message m, the model delivers the message to all neighbors in E and (perhaps) *some* neighbors in $E' \setminus E$. It then returns an *acknowledgment* of m to u indicating the broadcast is complete. These are the only message deliveries performed by the model. We assume nodes are well-formed in the sense that they always wait for the acknowledgment of their current message before initiating a new broadcast.

This model provides two timing bounds, defined with respect to two positive constants, F_{ack} and F_{prog} which are fixed for each execution. The first is the *acknowledgment* bound, which guarantees that each broadcast will complete and be acknowledged within F_{ack} time. The second is the *progress* bound, which guarantees the following slightly more complex condition: fix some $(u, v) \in E$ and interval of length F_{prog} throughout which u is broadcasting a message m; during this interval v must receive some message (though not necessarily m). The progress bound, in other words, bounds the time for a node to receive some message when at least one of its neighbors is broadcasting. As mentioned in the introduction, in both theory and practice it is reasonable to assume that F_{prog} is much smaller than F_{ack}. We emphasize that in abstract MAC layer models, the choice of which neighbors in $E' \setminus E$ receive a given message, as well as the order of receive events, are determined non-deterministically by an arbitrary *message scheduler*. The timing of these events is also determined non-deterministically by the scheduler, constrained only by the above time bounds.

We assume that nodes have unique ids. We also assume that each node can differentiate between their neighbors in E and $E' \setminus E$, an assumption justified by the standard practice in real networks of assessing link quality. For a given network definition (G, G'), we use D to describe the diameter of G, and $d_G(u, v)$, for $u, v \in V$, to describe the shortest path distance between u and v in G. We define D' and $d_{G'}$ similarly, but for G'. Finally, when proving lower bounds, we explicitly model randomness by passing each node at the beginning of the execution sufficiently many random bits to resolve probabilistic choices.

The Standard Abstract MAC Layer. The standard abstract MAC layer models nodes as event-driven automata. It assumes that an environment abstraction fires a *wake-up* event at each node at the beginning of each execution. The environment is also responsible for any events specific to the problem being solved. In multi-message broadcast, for example, the environment provides the broadcast messages to nodes at the beginning of the execution.

The Enhanced Abstract MAC Layer. The enhanced abstract MAC layer model differs from the standard model in two ways. First, it allows nodes access to time (formally, they can set timers that trigger events when they expire), and assumes nodes know F_{ack} and F_{prog}. Second, the model also provides nodes an *abort* interface that allows them to abort a broadcast in progress.

Restrictions on G'. When studying a problem in an abstract MAC layer model, it is often useful to consider constraints on the graph G'. In the original paper on these models [29], for example, we considered the very strong constraint that $G' = G$. In this paper, we consider three more general constraints on G'.

First, we say G' is *arbitrary* to indicate that we place no restrictions on its definitions (other than the default constraint of $E \subset E'$). Second, we say G' is *r-restricted*, for some $r \geq 1$, if for every $(u, v) \in E'$, $d_G(u, v) \leq r$. In studying this constraint, we sometimes use the notation G^r to describe the graph with edges between every $u, v \in V$, $u \neq v$, where u and v are within r hops in G. An r-restricted G' is a subgraph of G^r. Third, we say G' is *grey zone restricted* if (in addition to the general constraint of $E \subseteq E'$), the following is also true: we can embed the nodes in the Euclidean plane, giving each $v \in V$ a position $p(v) \in \mathbb{R}^2$ such that (1) For each pair of nodes $v, u \in V$, $(v, u) \in E$ if and only if $\|p(v) - p(u)\|_2 \leq 1$, and (2) for each pair of nodes $v, u \in V$, if $(v, u) \in E'$, then $\|p(v) - p(u)\|_2 \leq c$, where c is a universal constant such that $c \geq 1$. The range between 1 and c, in other words, describes a *grey zone* in which communication is uncertain. We emphasize that the second property described above only states that edges in E' cannot be longer than c, it does not require that every pair of nodes that have distance less than or equal to c must be G'-neighbors.

The Multi-Message Broadcast Problem. The multi-message broadcast (MMB) problem assumes the environment injects $k \geq 1$ messages into the network at the beginning of an execution,[4] perhaps providing multiple messages to the same node. We assume k is not known in advance. The problem is solved once every message m, starting at some node u, reaches every node in u's connected component in G. To achieve strong upper bound we do not, in other words, assume that G is connected. We treat messages as black boxes that cannot be combined; for example, we do not consider network coding solutions. We also assume that only a constant number of these messages can fit into a single local broadcast message. In this paper, we consider both deterministic and randomized algorithms. We require randomized solutions to solve the problem with high probability (w.h.p.), which we define to be at least $1 - 1/n$.

3 Multi-Message Broadcast with a Standard Abstract MAC Layer

In this section we study multi-message broadcast in what we call the *standard abstract MAC layer model*. As mentioned in the introduction, in previous work [29, 30] we described the *Basic Multi-Message Broadcast* (BMMB) algorithm, which implements the standard strategy of broadcasting each message only the first time you receive it. In more detail, the BMMB protocol works as follows. Every process i maintains a FIFO queue and list of received messages. When a process i receives a message m from the MAC layer it checks whether it already received it. If it has already received it, it discards the message. Otherwise, process i adds m to the back of its queue. Process i broadcasts the message at the head of its queue (if its queue is not empty) and waits for acknowledgment from the MAC layer. When the MAC layer acknowledges the message, i removes it from the queue and moves on to the next message (if any).

In [30], we proved that BMMB solves the MMB problem in $O(DF_{prog} + kF_{ack})$ time under the strong assumption that $G' = G$. In the two subsections that follow, we study its behavior under more general definitions of G'. We then prove a lower bound for all MMB solutions.

3.1 The BMMB Algorithm for Arbitrary G'

The natural next step in analyzing BMMB is considering its performance in our model when we assume an arbitrary G'. It is easy to show, of course, that the algorithm always terminates in $O(DkF_{ack})$ time, as a message m, on arriving at a new hop, must wait for at most $O(k)$ messages to be broadcast before it too is broadcast. Here we apply a more detailed pipeline argument to show that BMMB performs better than this basic bound in this difficult setting.

Theorem 3.1. *The BMMB algorithm solves the MMB problem in $O((D + k)F_{ack})$ time in the standard abstract MAC layer model for arbitrary G'.*

Proof. For the sake of analysis, assume each node u keeps a *sent* set to which it adds every message that it has successfully broadcast (i.e., broadcast and received an ack for). Next, fix some execution and an arbitrary message m from among the k messages provided to BMMB to broadcast in this

[4] A general version of the MMB problem, in which the messages arrive in an online manner, is studied in [30] and elsewhere.

execution. Let $u_m \in V$ be the node that is provided m by the environment at the start of the execution. For each node $v \in V$, let $d_v = d_G(v, u_m)$. For each $\ell \in [1, k]$, let $t_\ell(v) = d_v + \ell$.

Our strategy is to prove the following key claim: *for each $\ell \in [1, k]$ and node v, after $t_\ell(v)F_{ack}$ time, node v's sent set either: (1) contains m; or (2) contains at least ℓ other messages.* Once we prove this claim, it will then follow that after $t_k(v)F_{ack} \leq (D + k)F_{ack}$ time, v has sent (and thus also received) m. Applying this to all nodes and all k messages then yields the theorem statement. We prove our key claim using induction on $h = d_v + \ell$. For the base case of $h = 0$, notice that $h = 0$ implies $d_v = 0$. This, in turn, reduces the key claim to a statement about the local queue of v that follows directly from the definition of the algorithm.

For the inductive step, consider some v such that $d_v + \ell = h$. To show the key claim holds for $t_\ell(v) = h$, we leverage the inductive hypothesis for nearby nodes and combinations of relevant values that sum to $h - 1$. First, we note that if $d_v = 0$, then the base case argument once again applies. Assume, therefore, that $d_v \geq 1$. Next, fix some G-neighbor u of v such that $d_u = d_v - 1$. By the induction hypothesis, we know that after $s = t_{\ell-1}(v)F_{ack} = t_\ell(u)F_{ack}$ time: v has either sent m or sent $\ell - 1$ other messages, and u has either sent m or sent ℓ other messages. If v has sent m or at least ℓ messages by s then we are done. If it has not, then by time s, u will have either sent it m or a new message (i.e., distinct from the $\ell - 1$ messages v has already sent by time s). In either case, in the F_{ack} time that passes between s and $t_\ell(v)F_{ack}$, v will either send m or an ℓ^{th} message, establishing the key claim for $t_\ell(v) = h$. □

3.2 The BMMB Algorithm for r-Restricted G'

We have shown that moving from $G' = G$ to an unrestricted definition of G' slows down the performance of BMMB; i.e., replacing a DF_{prog} factor with DF_{ack}, which might be substantially slower. In this section we seek a middle ground—attempting to identify just enough constraints on G' to enable faster MMB performance. In more detail, we study BMMB with an r-restricted G' and prove that its performance scales well with r:

Theorem 3.2. *For any $r \geq 1$, the BMMB algorithm solves the MMB problem in $O(DF_{prog} + rkF_{ack})$ time in the standard abstract MAC layer model with an r-restricted G'.*

Notice that for small r, this bound is better than the $O((D + k)F_{ack})$ bound we established in Theorem 3.1 for arbitrary G', and comes close to matching the bound proved in [30] for the case where $G' = G$. This result implies the DF_{ack} factor of the arbitrary G' bound is somehow dependent on the possibility of G' edges between nodes distant in G. We emphasize that the above bound does not follow naturally from the $O(DF_{prog} + kF_{ack})$ bound for $G' = G$. The presence of unreliability fundamentally disrupts the core induction of this existing bound and requires a new approach.

We present the proof of Theorem 3.2 using a more formal description of the abstract MAC layer. We can model our systems using *Timed I/O Automata* [21].

Consider some positive integer r. The r'th power $G^r(V, E^r)$ of a graph G is a graph with the same vertex set V, and two nodes $v, u \in V$ are adjacent, $(u, v) \in E^r$, when their distance in G is at most r. That is, $E^r = \{(u, v) \mid u \neq v \text{ and } d_G(u, v) \leq r\}$. (The rth power graph G^r does not includes self-loops.) For a given node $j \in V$, let $N^r_G(j) = \{j' \mid d_G(j, j') \leq z\}$ denote the set of nodes that are within r hops of j in G, including j itself. A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ (denoted by $G \subseteq G'$), if $V = V'$ and $E' \subseteq E$.

3.2.1 Guarantees for the Abstract MAC Layer

Here we provide a set of properties that constrain the behavior of the abstract MAC layer automaton. Technically, these properties are expressed for admissible timed executions of the timed I/O automaton modeling the complete system.

Well-Formedness Properties. We assume some constraints on the behavior of the user automata. Let α be an admissible execution of the system consisting of user and abstract MAC layer automata. We say α is *user-well-formed* if the following hold, for every i:

1. Every two $bcast_i$ events have an intervening ack_i or $abort_i$ event.
2. Every $abort(m)_i$ is preceded by a $bcast(m)_i$ (for the same m) with no intervening $bcast_i$, ack_i, or $abort_i$ events.

The rest of this subsection gives constraints on the behavior of the abstract MAC layer automaton, in system executions that are user-well-formed. Thus, from now on in the section, we assume that α is a user-well-formed system execution.

Constraints on Message Behavior. We assume that there exists a "cause" function that maps every $rcv(m)_j$ event in α to a preceding $bcast(m)_i$ event, where $i \neq j$, and that maps each $ack(m)_i$ and $abort(m)_i$ to a preceding $bcast(m)_i$.

We use the term *message instance* to refer to the set consisting of a *bcast* event and all the other events that are related to it by the *cause* function.

We now define two safety conditions and one liveness condition regarding the relationships captured by the cause function:

1. **Receive correctness:** Suppose that $bcast(m)_i$ event π causes $rcv(m)_j$ event π' in α. Then $(i,j) \in E'$, and no other $rcv(m)_j$ event or $ack(m)_i$ event caused by π precedes π'.
2. **Acknowledgment correctness:** Suppose that $bcast(m)_i$ event π causes $ack(m)_i$ event π' in α. Then for every j such that $(i,j) \in E$, a $rcv(m)_j$ event caused by π precedes π'. Also, no other $ack(m)_i$ event or $abort(m)_i$ caused by π precedes π'.
3. **Termination:** Every $bcast(m)_i$ causes either an $ack(m)_i$ or an $abort(m)_i$.

Time Bounds. We now impose upper bounds on the time from a $bcast(m)_i$ event to its corresponding $ack(m)_i$ and $rcv(m)_j$ events.[5]

Let F_{ack} and F_{prog} be positive real numbers. We use these to bound delays for a specific message to be delivered and an acknowledgment received (the "acknowledgment delay"), and for *some* message from among many to be received (the "progress delay"). We think of F_{prog} as smaller than F_{ack}, because the time to receive *some* message among many is typically less than the time to receive a specific message. The statement of the acknowledgment bound is simple:

4. **Acknowledgment bound:** Suppose that a $bcast(m)_i$ event π causes an $ack(m)_i$ event π' in α. Then the time between π and π' is at most F_{ack}.

[5] We express these here as constants rather than functions, because we will not worry about adaptive bounds in this paper.

The statement of the progress bound is a bit more involved, and requires some auxiliary definitions. Let α' be a closed execution fragment within the given execution α,[6] and let j be a process. Then define:

- *connect*(α', j) is the set of message instances in α such that α' is wholly contained between the *bcast* and terminating event (*ack* or *abort*) of the instance, and $(i, j) \in E$, where i is the originator of the *bcast* of the message instance.

- *contend*(α', j) is the set of message instances in α for which the terminating event does not precede the beginning of α', and $(i, j) \in E'$, where i is the originator of the *bcast* of the message instance.

Lemma 3.3. *For every α' and j, connect$(\alpha', j) \subseteq$ contend(α', j).*

5. **Progress bound:** For every closed fragment α' within α, and for every process j, it is not the case that all three of the following conditions hold:

   (a) The total time of α' is strictly greater than F_{prog}.

   (b) *connect*$(\alpha', j) \neq \emptyset$.

   (c) No rcv_j event from a message instance in *contend*(α', j) occurs by the end of α'.

   In other words, j should receive *some* message within time F_{prog} provided that at least one message is being sent by a G-neighbor.

Note that our definitions allow a *rcv* for a particular *bcast* to occur after an *abort* for that *bcast*. We impose a (small) bound ϵ_{abort} on the amount of time after an *abort* when such a *rcv* may occur.

3.2.2 The Multi-Message Broadcast Problem

A user automaton is considered to be an *MMB protocol* provided that its external interface includes an $arrive(m)_i$ input and $deliver(m)_i$ output for each user process i and message $m \in \mathcal{M}$.

We say an execution of an MMB protocol is *MMB-well-formed* if and only if it contains at most one $arrive(m)_i$ event for each $m \in \mathcal{M}$; that is, each broadcast message is unique. We say an MMB protocol *solves the MMB problem* if and only if for every MMB-well-formed (admissible) execution α of the MMB protocol composed with a MAC layer, the following hold:

(a) For every $arrive(m)_i$ event in α and every process j, α contains a $deliver(m)_j$ event.

(b) For every $m \in \mathcal{M}$ and every process j, α contains at most one $deliver(m)_j$ event and it comes after an $arrive(m)_i$ event for some i.

We describe a simple MMB protocol that achieves efficient runtime.

The Basic Multi-Message Broadcast (BMMB) Protocol
Every process i maintains a FIFO queue named *bcastq* and a set named *rcvd*. Both are initially empty.

[6] Formally, that means that there exist fragments α'' and α''' such that $\alpha = \alpha''\alpha'\alpha'''$, and moreover, the first state of α' is the last state of α''. Notice, this allows α' to begin and/or end in the middle of a trajectory.

If process i is not currently sending a message (i.e., not waiting for an *ack* from the MAC layer) and *bcastq* is not empty, the process immediately (without any time-passage) *bcast*s the message at the head of *bcastq* on the MAC layer.

When process i receives an $arrive(m)_i$ event, it immediately performs a local $deliver(m)_i$ output and adds m to the back of its *bcastq*, and to its *rcvd* set.

When i receives a message m from the MAC layer it checks its *rcvd* set. If $m \in rcvd$, procsss i discards the message. Otherwise, i immediately performs a $deliver(m)_i$ event, and adds m to the back of its *bcastq* and to its *rcvd* set.

Theorem 3.4. *The BMMB protocol solves the MMB problem.*

We give two definitions that we will use in our complexity proof. In the following, let α be some MMB-well-formed execution of the BMMB protocol composed with a MAC layer. We begin with two definitions that we will use in our complexity proof.

***get* events.** We define a $get(m)_i$ event with respect to α, for some arbitrary message m and process i, to be one in which process i first learns about message m. Specifically, $get(m)_i$ is the first $arrive(m)_i$ event in case message m arrives at process i, otherwise, $get(m)_i$ is the first $rcv(m)_i$ event.

***clear* events** Let $m \in \mathcal{M}$ be a message for which an $arrive(m)_i$ event occurs in α. We define $clear(m)$ to describe the final $ack(m)_j$ event in α for any process j.[7]

3.2.3 Proof Preliminaries

For the rest of Section 3, we consider the special case of the general MMB problem in which all messages arrive from the environment at time $t_0 = 0$, that is, all *arrive* events occur at time 0. We fix a particular MMB-well-formed execution α of the BMMB protocol composed with a MAC layer. We assume that an $arrive(m)_{i_0}$ event occurs in α for some message $m \in \mathcal{M}$ at some node i_0, at time $t_0 = 0$.

For each node $i \in V$ and each time t, we introduce two sets of messages, which we call $\mathcal{R}$ (for ”received messages”) and $\mathcal{C}$ (for ”completed messages”). We define:

- $\mathcal{R}_i(t) \subseteq \mathcal{M}$ is the set of messages $m' \in \mathcal{M}$ for which the $get(m')_i$ event occurs by time t.
- $\mathcal{C}_i(t) \subseteq \mathcal{M}$ is the set of messages $m' \in \mathcal{M}$ for which the $ack(m')_i$ event occurs by time t.

That is, $\mathcal{R}_i(t)$ is the set of messages that have been received by process i by time t, and $\mathcal{C}_i(t)$ is the set of messages that process i has finished (completed) processing by time t.[8]

The following two lemmas express some very basic properties of the $\mathcal{R}$ and $\mathcal{C}$ sets.

Lemma 3.5. *For every i, i', t and t' such that $t \leq t'$:*

1. $\mathcal{R}_i(t) \subseteq \mathcal{R}_i(t')$ *and* $\mathcal{C}_i(t) \subseteq \mathcal{C}_i(t')$.

[7] By the definition of BMMB, if an $arrive(m)_i$ occurs, then i eventually *bcasts* m, so $ack(m)_i$ eventually occurs. Furthermore, by the definition of BMMB, there can be at most one $ack(m)_j$ event for every process j. Therefore, $clear(m)$ is well-defined.

[8] Note that both $\mathcal{R}_i(t)$ and $\mathcal{C}_i(t)$ may include m.

2. *$\mathcal{C}_i(t) \subseteq \mathcal{R}_i(t')$.*

3. *If i and i' are neighbors in G, then $\mathcal{C}_{i'}(t) \subseteq \mathcal{R}_i(t')$.*

Proof. Straightforward. □

Lemma 3.6. *Fix i and $t \geq 0$, and let s be the final state at time t. Then, in state s, $bcastq_i$ contains exactly the messages in $\mathcal{R}_i(t) - \mathcal{C}_i(t)$.*

Proof. By the operation of the algorithm, $bcastq_i$ contains exactly the messages that have had a *get* and no *ack* at node i. These are exactly the elements of $\mathcal{R}_i(t) - \mathcal{C}_i(t)$. □

Lemma 3.7. *Fix i and $t \geq 0$, and let s be the final state at time t.*

1. *If in state s, m' is in position $k \geq 1$ of $bcastq_i$, then $m' \in \mathcal{C}_i(t + kF_{ack})$.*

2. *If in state s, $bcastq_i$ has length at least $k \geq 0$, then $|\mathcal{C}_i(t + kF_{ack})| \geq |\mathcal{C}_i(t)| + k$.*

3. *If $|\mathcal{R}_i(t)| \geq k \geq 0$, then $|\mathcal{C}_i(t + kF_{ack})| \geq k$.*

Proof. 1. For $k = 1$, the abstract MAC layer properties say that, within time F_{ack}, m' is acknowledged at i. Therefore, $m' \in \mathcal{C}_i(t+F_{ack})$, which yields the result for $k = 1$. The statement for general k follows from repeated application of the statement for $k = 1$.

2. The statement is trivial for $k = 0$, so consider $k \geq 1$. Part 1, applied to the first k messages in $bcastq_i$ in state s, implies that all of these messages are in $\mathcal{C}_i(t+F_{ack})$. Lemma 3.6, implies that none of these messages are in $\mathcal{C}_i(t)$. Therefore, $|\mathcal{C}_i(t + F_{ack})| \geq |\mathcal{C}_i(t)| + k$.

3. Suppose that $|\mathcal{R}_i(t)| = |\mathcal{C}_i(t)| + |\mathcal{R}_i(t) - \mathcal{C}_i(t)| \geq k$. By Lemma 3.6, every element of $\mathcal{R}_i(t) - \mathcal{C}_i(t)$ is on $bcastq_i$ in state s, so the length of $bcastq_i$ in state s is at least $|\mathcal{R}_i(t) - \mathcal{C}_i(t)|$. We consider two cases. If $|\mathcal{R}_i(t) - \mathcal{C}_i(t)| \leq k$, then

$$|\mathcal{C}_i(t + k \cdot F_{ack})| \geq |\mathcal{C}_i(t + |\mathcal{R}_i(t) - \mathcal{C}_i(t)| \cdot F_{ack})| \geq |\mathcal{C}_i(t)| + |\mathcal{R}_i(t) - \mathcal{C}_i(t)| \geq k,$$

as needed for Part 3, where the first inequality follows, since $\mathcal{C}_i(t + |\mathcal{R}_i(t) - \mathcal{C}_i(t)| \cdot F_{ack}) \subseteq \mathcal{C}_i(t + k \cdot F_{ack})$ by Part 1 of Lemma 3.5 (with $t = t + |\mathcal{R}_i(t) - \mathcal{C}_i(t)| \cdot F_{ack}$ and $t' = t + k \cdot F_{ack}$); the second inequality follows by Part 2 of the lemma; and the last inequality holds by the assumption of Part 3 (of the lemma). If $|\mathcal{R}_i(t) - \mathcal{C}_i(t)| > k$, then $|\mathcal{C}_i(t + k \cdot F_{ack})| \geq |\mathcal{C}_i(t)| + k$, by Part 2 of the lemma. Part 3 follows.

□

The following corollary is a special case of Part 2 of Lemma 3.7.

Corollary 3.8. *Fix i, $t \geq 0$ and $\ell > 0$. If $|\mathcal{C}_i(t)| \geq \ell - 1$ and $|\mathcal{R}_i(t)| \geq \ell$, then $|\mathcal{C}_i(t + F_{ack})| \geq \ell$.*

Proof. The case where $|\mathcal{C}_i(t)| \geq \ell$ follows by Part 1 of Lemma 3.5. Suppose that $|\mathcal{C}_i(t)| = \ell - 1$. Since $|\mathcal{R}_i(t)| \geq \ell$, it follows that $\mathcal{R}_i(t) - \mathcal{C}_i(t) \neq \emptyset$. Therefore, by Lemma 3.6, at the final state s at time t, $bcastq_i$ has length at least at least one. Thus, by Part 2 of Lemma 3.7, $|\mathcal{R}_i(t + F_{ack})| \geq |\mathcal{R}_i(t)| + 1$. The Corollary follows. □

Now we have two key lemmas that describe situations when a process i is guaranteed to receive a new message. The first deals with i receiving its first message.[9]

Lemma 3.9. *Let i and j be neighboring nodes in G, and suppose $t \geq 0$. If $\mathcal{R}_j(t) \neq \emptyset$, then $\mathcal{R}_i(t + F_{prog}) \neq \emptyset$.*

Proof. Assume for contradiction that $\mathcal{R}_i(t + F_{prog}) = \emptyset$. Choose $t' > t + F_{prog}$ to be some time strictly after $t + F_{prog}$, when $\mathcal{R}_i(t') = \emptyset$; this is possible because the next discrete event after time $t + F_{prog}$ must occur some positive amount of time after $t + F_{prog}$.

We obtain a contradiction to the progress bound. Let α' be the closed execution fragment of α that begins with the final state s at time t and ends with the final state s' at time t'. We show that α' provides a contradiction to the progress bound. We verify that the three conditions in the definition of the progress bound are all satisfied for α'. Condition (a), that the total time of α' is strictly greater than F_{prog}, is immediate.

Condition (b) says that $\mathit{connect}(\alpha', i) \neq \emptyset$. Since $\mathcal{R}_i(t) = \emptyset$, Lemma 3.5, Part 3, implies that $\mathcal{C}_j(t) = \emptyset$. Since $\mathcal{R}_j(t) \neq \emptyset$, Lemma 3.6, implies that, in state s, bcastq_j is nonempty. Let m' be the message at the head of bcastq_j in state s. Since s is the final state at time t and the protocol has 0 delay for performing *bcast*s, it must be that the *bcast* event for process j's instance for m' occurs before the start of α'. Since $m' \notin \mathcal{R}_i(t')$, m' is not received by process i by the end of α'. This implies that the $\mathit{ack}_j(m')$ event, which terminates j's instance for m', must occur after the end of α'. It follows that j's instance for m' is in $\mathit{connect}(\alpha', i)$, so that $\mathit{connect}(\alpha', i) \neq \emptyset$, which shows Condition (b).

Condition (c) says that no rcv_i event from a message instance in $\mathit{contend}(\alpha', i)$ occurs by the end of α'. We know that no rcv_i occurs by the end of α', because $\mathcal{R}_i(t') = \emptyset$. So no rcv_i event from a message instance in $\mathit{contend}(\alpha', i)$ occurs by the end of α', which shows Condition (c).

Thus, α' satisfies the combination of three conditions that are prohibited by the progress bound assumption, yielding the needed contradiction. □

The next lemma deals with the fast positive progress scenario in which process j receives some "new" message in F_{prog} time.

Lemma 3.10. *Let i and j be neighboring nodes in G, and suppose $t \geq 0$. Suppose that:*

1. *$\mathcal{R}_i(t) \subseteq \mathcal{C}_{i'}(t)$ for every neighbor i' of i in G'.*
2. *$\mathcal{R}_j(t) - \mathcal{R}_i(t) \neq \emptyset$.*

Then, $|\mathcal{R}_i(t + F_{prog})| > |\mathcal{R}_i(t)|$.

This says that, if every message that i has already received is already completed at all of i's neighbors in G' and some neighbor j of i in G has received some message that i hasn't yet received, then i will receive a new message within F_{prog} time.

Proof. Assume for contradiction that $\mathcal{R}_i(t) \subseteq \mathcal{C}_{i'}(t)$ for every neighbor i' of i in G', that $\mathcal{R}_j(t) - \mathcal{R}_i(t) \neq \emptyset$, and that $|\mathcal{R}_i(t + F_{prog})| = |\mathcal{R}_i(t)|$. Then it must be that $\mathcal{R}_i(t + F_{prog}) = \mathcal{R}_i(t)$. Choose $t' > t + F_{prog}$ to be some time strictly after $t + F_{prog}$, when $\mathcal{R}_i(t') = \mathcal{R}_i(t)$; this is possible because the next discrete event after time $t + F_{prog}$ must occur some positive amount of time after $t + F_{prog}$.

[9] Actually, this lemma is formally a corollary to the following one, but it might be nicer to see this proof as a "warm-up".

We obtain a contradiction to the progress bound. Let α' be the closed execution fragment of α that begins with the final state s at time t and ends with the final state s' at time t'. We verify that the three conditions in the definition of the progress bound are all satisfied for α'.

- Condition (a): The total time of α' is strictly greater than F_{prog}.
  This is immediate.

- Condition (b): $connect(\alpha', i) \neq \emptyset$.
  Since $\mathcal{R}_j(t) - \mathcal{R}_i(t) \neq \emptyset$ and $\mathcal{C}_j(t) \subseteq \mathcal{R}_i(t)$ (by Part 3 of Lemma 3.5 with $i = i$, $i' = j$), we have $\mathcal{R}_j(t) - \mathcal{C}_j(t) \neq \emptyset$.

  Then Lemma 3.6, implies that, in state s, $bcastq_j$ is nonempty. Let m' be the message at the head of $bcastq_j$ in state s. Since s is the final state at time t and the protocol has 0 delay for performing *bcast*s, it must be that the *bcast* event for process j's instance for m' occurs before the start of α'.

  Also, we know that $m' \notin \mathcal{R}_i(t)$, because $m' \notin \mathcal{C}_j(t)$ and $\mathcal{R}_i(t) \subseteq \mathcal{C}_j(t)$.

  Since $\mathcal{R}_i(t') = \mathcal{R}_i(t)$, we also know that $m' \notin \mathcal{R}_i(t')$. Therefore, m' is not received by process i by the end of α'. This implies that the $ack_j(m')$ event, which terminates j's instance for m', must occur after the end of α'. It follows that j's instance for m' is in $connect(\alpha', i)$, so that $connect(\alpha', i) \neq \emptyset$.

- Condition (c): No rcv_i event from a message instance in $contend(\alpha', i)$ occurs by the end of α'. We claim that, if a message m'' has an instance in $contend(\alpha', i)$, then $m'' \notin \mathcal{R}_i(t)$. To see this, let i' be a neighbor of i in G' originating an instance of m'' in $contend(\alpha', i)$. If $m'' \in \mathcal{R}_i(t)$, then by hypothesis 1 (of the lemma), also $m'' \in \mathcal{C}_{i'}(t)$. That means that the *ack* event of node i''s instance of m'' occurs before the start of α', which implies that the instance is not in $contend(\alpha, i)$.

  With this claim, we can complete the proof for Condition (c). Suppose for contradiction that a $rcv_i(m'')$ event from some message instance in $contend(\alpha, i)$ occurs by the end of α'. Using the first claim above, $m'' \in \mathcal{R}_i(t')$. But by the second claim above, $m'' \notin \mathcal{R}_i(t)$. But we have assumed that $\mathcal{R}_i(t') = \mathcal{R}_i(t)$, which yields a contradiction.

Thus, α' satisfies the combination of three conditions that are prohibited by the progress bound assumption, yielding the needed contradiction. □

The next lemma deals with the slow positive progress scenario in which process j is guaranteed to receive some "new" message in $z \cdot F_{ack}$ time (for some positive integer z).

Lemma 3.11. *Fix some time $t \geq 0$. Suppose that:*

1. $|\mathcal{C}_j(t)| \geq \ell - 1$.

2. $\mathcal{C}_j(t) \subseteq \mathcal{C}_{j'}(t)$ *for every node* $j' \in N_G^z(j)$.

3. *There exists some* $j'' \in N_G^z(j)$ *such that* $|\mathcal{R}_{j''}(t)| \geq \ell$.

Then $|\mathcal{R}_j(t + zF_{ack})| \geq \ell$.

This says that, if (1) j completes at least $\ell - 1$ messages by time t; (2) every message that j has completed by time t is also completed at all of j's neighbors in G^z by time t; and (3) there exists at least one neighbor of j in G^z that receives at least ℓ messages by that time, then j receives at least ℓ messages by time $t + z \cdot F_{ack}$.

Proof. We prove this lemma by induction on z. For the base, $z = 0$, the statement trivially follows, since $N^0(j) = \{j\}$, which implies together with condition 3 that $|\mathcal{R}_j(t)| \geq \ell$. For the inductive step, we assume $z \geq 1$. We assume the lemma statement for $z' < z$ and prove it for z. If $|\mathcal{R}_j(t)| \geq \ell$, then by Part 1 of Lemma 3.5, $|\mathcal{R}_j(t + F_{ack})| \geq \ell$ and we are done.

If $\mathcal{C}_j(t+F_{ack}) \neq \mathcal{C}_j(t)$, then by assumption 1 and Part 1 of Lemma 3.5, $|\mathcal{C}_j(t+F_{ack})| \geq \ell$, which implies by Part 2 of Lemma 3.5 that $|\mathcal{R}_j(t + z \cdot F_{ack})| \geq \ell$, as needed.

It remains to consider the case where $|\mathcal{R}_j(t)| = \ell - 1$ and $\mathcal{C}_j(t + F_{ack}) = \mathcal{C}_j(t)$. Let j'' be a closest neighbor of j such that $|\mathcal{R}_{j''}(t)| \geq \ell$. Note that, j'' must be at distance at least 1 from j (by the assumptions for this case) and j'' must be at distance at most z from j (by assumption 3 of the lemma). Moreover, by the first two assumptions of the lemma, it follow that $|\mathcal{C}_{j''}(t)| \geq \ell - 1$. Combining this inequality with $|\mathcal{R}_{j''}(t)| \geq \ell$ and Corollary 3.8, we get that

$$|\mathcal{C}_{j''}(t + F_{ack})| \geq \ell. \tag{1}$$

Let j^* be the next-closer node on a shortest path in G from j'' to j. We now apply the inductive hypothesis for $z' = z - 1$ and time $t' = t + F_{ack}$. Note that, $j^* \in N_G^{z'}(j)$. To do this, we show that the three assumptions of the lemma indeed hold for z' and t'. The first assumption of the lemma holds as

$$|\mathcal{C}_j(t')| \geq |\mathcal{C}_j(t)| \geq \ell - 1,$$

where the first inequality holds since $\mathcal{C}_j(t) \subseteq \mathcal{C}_j(t')$ (by Part 1 of Lemma 3.5, with $t \leq t'$); and the second inequality holds by assumption 1 for z and t. The second assumption of the lemma holds as,

$$\mathcal{C}_j(t') \subseteq \mathcal{C}_{j'}(t) \subseteq \mathcal{C}_{j'}(t'), \text{ for every node } j' \in N_G^{z'},$$

where the first inequality holds since, in this case, $\mathcal{C}_j(t') = \mathcal{C}_j(t)$ and $\mathcal{C}_j(t) \subseteq \mathcal{C}_{j'}(t)$ (by assumption 2 of the lemma for $z > z'$ and t); and the second inequality holds by Part 1 of Lemma 3.5 with $t \leq t'$. We next argue that the third assumption holds as well. Specifically, we claim that j^*, in particular, satisfies this assumption. That is, $j^* \in N_G^{z'}(j)$ and $|\mathcal{R}_{j^*}(t')| \geq \ell$. The first statement holds, since $z' = z - 1$ and $d_G(j, j^*) < d_G(j, j'') \leq z$. The second statement holds as,

$$|\mathcal{R}_{j^*}(t')| \geq |\mathcal{C}_{j''}(t')| \geq \ell,$$

where the first inequality holds, since $\mathcal{C}_{j''}(t') \subseteq \mathcal{R}_{j^*}(t')$ (by Part 3 of Lemma 3.5) and the second inequality holds by combining together Inequality (1) with $t' = t + F_{ack}$.

Having shown the three assumptions, we can now invoke the inductive hypothesis for z' and t'. We have $|\mathcal{R}_j(t' + z' \cdot F_{ack})| \geq \ell$. In addition, $t' + z' \cdot F_{ack} \leq t + z \cdot F_{ack}$, since $t' = t + F_{ack}$ and $z' \leq z - 1$. Combining these two inequalities together with Part 1 of Lemma 3.5, we get that $|\mathcal{R}_j(t + z \cdot F_{ack})| \geq \ell$, as needed. The lemma follows. □

3.2.4 The Key Lemma

We continue to assume all the context we established earlier in Subsection 3.2.3. The lemma below summarizes some helpful complexity bounds.

Lemma 3.12 ("Complexity Bounds")**.** *The following hold for the $t_{d,\ell}$ values:*

1. *For $d' \leq d''$, $t_{d',\ell} \leq t_{d'',\ell}$ (monotonically increasing in terms of d).*
2. *For $\ell \geq 2$, $d \geq 1$, $t_{d+r,\ell-1} + r \cdot F_{ack} \leq t_{d,\ell}$.*
3. *For $\ell \geq 2$, $d \geq 1$, $t_{d+r,\ell-1} + F_{ack} \leq t_{d-1,\ell}$.*
4. *For $\ell \geq 1$, $d \geq 1$, $t_{d-1,\ell} + F_{prog} = t_{d,\ell}$.*
5. *For $\ell > 1$, $\ell \cdot F_{ack} \leq t_{0,\ell}$.*

Proof. By simple algebraic calculations. □

To prove the key lemma, we show a double induction for ℓ as an "outer" induction and for distance d as an "inner" induction. To warm up, let us begin with two special cases. The first (Lemma 3.13 below) will be used in the base case for $\ell = 1$ for the outer induction of the inductive proof in the main lemma. The second (see Lemma 3.14) will be used in the base case for $d = 0$ for the inner induction of the inductive proof in the main lemma.

Lemma 3.13. *Let j be a node at distance $d = d_G(i_0, j)$ from i_0. Then:*

1. *$\mathcal{R}_j(t_{d,1}) \neq \emptyset$.*
2. *$\mathcal{C}_j(t_{d,1} + F_{ack}) \neq \emptyset$.*

Proof.

1. For Part 1, we use induction on d. For the base case, consider $d = 0$. Then $j = i_0$ and $t_{d,1} = t_{0,1} = 0$. Since $m \in \mathcal{R}_{i_0}(0)$, we see that $\mathcal{R}_j(t_{d,1}) \neq \emptyset$, as needed.

   For the inductive step, assume Part 1 for $d-1$ and prove it for d. Let j' be the predecessor of j on a shortest path in G from i_0 to j; then $d_G(i_0, j') = d - 1$. By inductive hypothesis, we know that $\mathcal{R}_{j'}(t_{d-1,1}) \neq \emptyset$. Then Lemma 3.9 implies that $\mathcal{R}_j(t_{d-1,1} + F_{prog}) \neq \emptyset$. Since $t_{d,1} = t_{d-1,1} + F_{prog}$, this implies that $\mathcal{R}_j(t_{d,1}) \neq \emptyset$, as needed.
2. Part 2 follows from Part 1 using Lemma 3.7, Part 3, applied with $k = 1$.

□

Lemma 3.14. *Let $\ell \geq 1$. Then:*

1. *$m \in \mathcal{R}_{i_0}(t_{0,\ell})$.*
2. *Either $m \in \mathcal{C}_{i_0}(t_{0,\ell} + F_{ack})$ or $|\mathcal{C}_{i_0}(t_{0,\ell} + F_{ack})| \geq \ell$.*

Proof. Since $m \in \mathcal{R}_{i_0}(0)$, and $0 \leq t_{0,\ell}$, we have $m \in \mathcal{R}_j(t_{0,\ell})$, which yields Part 1.

For Part 2, if $m \in \mathcal{C}_{i_0}(0)$, then clearly $m \in \mathcal{C}_{i_0}(t_{0,\ell} + F_{ack})$, which suffices. So suppose that $m \notin \mathcal{C}_{i_0}(0)$. Then $m \in \mathcal{R}_{i_0}(0) - \mathcal{C}_{i_0}(0)$, so m is on $bcastq_{i_0}$ in the final state s_0 at time $t = 0$.

If in state s_0, the position of m on $bcastq_{i_0}$ is $\leq \ell$, then Lemma 3.7, Part 1, implies that $m \in \mathcal{C}_{i_0}(\ell F_{ack})$. By Part 5 of Lemma 3.12, $\ell F_{ack} \leq t_{0,\ell} + F_{ack}$, which implies together with Part 1 of Lemma 3.5 that $m \in \mathcal{C}_{i_0}(t_{0,\ell} + F_{ack})$, which is sufficient to establish the claim.

On the other hand, if in state s_0, the position of m on $bcastq_{i_0}$ is strictly greater than ℓ, then we apply Lemma 3.7, Part 2, to conclude that $|\mathcal{C}_{i_0}(\ell F_{ack})| \geq \ell$. That implies that $|\mathcal{C}_{i_0}(t_{0,\ell} + F_{ack})| \geq \ell$, which again suffices. □

And now, for the main lemma.

Lemma 3.15. *Let j be a node at distance $d = d_G(i_0, j)$ from i_0 in G. Let ℓ be any positive integer. Then:*

1. *Either $m \in \mathcal{R}_j(t_{d,\ell})$ or $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$.*

2. *Either $m \in \mathcal{C}_j(t_{d,\ell} + F_{ack})$ or $|\mathcal{C}_j(t_{d,\ell} + F_{ack})| \geq \ell$.*

Proof. We prove both parts together by induction on ℓ. For the base, $\ell = 1$, both statements follow immediately from Lemma 3.13. For the inductive step, let $\ell \geq 2$. We assume the lemma statement for $\ell - 1$ (and for all d) and prove it for ℓ. To prove the lemma statement for ℓ, we use a second, "inner"induction, on the distance d from i_0 and the destination j. For the base, $d = 0$, both statements follow from Lemma 3.14.

Inductive Step: $d \geq 1$. For the inductive step, we assume $d \geq 1$. Assume both parts of the lemma for (1) $\ell - 1$ (as "outer" induction hypothesis) for all distances; and (2) for ℓ for distance $d - 1$ (as "inner" induction hypothesis). We prove both parts of the lemma for ℓ and distance d.

By our "outer" inductive hypothesis, all processors of the network satisfy the two parts of the lemma for $\ell - 1$ and all values of d. In particular, by combining the inductive hypothesis for $\ell - 1$ and all values of d with Part 1 of Lemma 3.12 and Part 1 of Lemma 3.5, it follows that for every node $j' \in N_G^r(j)$, either

(S1) $m \in \mathcal{C}_{j'}(t_{d+r,\ell-1} + F_{ack})$ or $|\mathcal{C}_{j'}(t_{d+r,\ell-1} + F_{ack})| \geq \ell - 1$.

We use distance $d + r$ for j' because j' is at distance at most $d + r$ from i_0 in G (j is at distance d from i_0 in G; and j' is either j itself or it at distance at most r from j in G, since $j' \in N_G^r(j)$). Let $t^* = t_{d+r,\ell-1} + F_{ack}$. Recall that, by Part 2 of Lemma 3.12,

$$t^* + (r-1)F_{ack} \leq t_{d,\ell} \ . \tag{2}$$

1. We now prove Part 1 of the lemma (for ℓ and d). Suppose that $m \in \mathcal{C}_j(t^*)$ or $|\mathcal{C}_j(t^*)| \geq \ell$. Then, either $m \in \mathcal{R}_j(t_{d,\ell})$ or $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$, since $\mathcal{C}_j(t^*) \subseteq \mathcal{R}_j(t_{d,\ell})$, by Inequality (2) and Part 2 of Lemma 3.5 (with $t = t^*$ and $t' = t_{d,\ell}$). This implies that j satisfies Part 1 of the lemma statement for ℓ. This implies Part 1. Now, suppose the contrary, that $m \notin \mathcal{C}_j(t^*)$ and $|\mathcal{C}_j(t^*)| < \ell$. Since, j does satisfy (S1) for $\ell - 1$, it follows that $|\mathcal{C}_j(t^*)| \geq \ell - 1$, since $m \notin \mathcal{C}_j(t^*)$. Thus, in the remaining case, we have

   $$m \notin \mathcal{C}_j(t^*) \text{ and } |\mathcal{C}_j(t^*)| = \ell - 1. \tag{3}$$

   We next prove that $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$ (which implies Part 1 of the lemma). We consider two cases regarding the set of messages that j completes by time t^* and the sets of messages that are completed (by that time) by all neighbors of j in G^r.

Case 1: There exists some neighbor j' of j in G^r such that $\mathcal{C}_{j'}(t^*) \neq \mathcal{C}_j(t^*)$.

Choose a closest node j'' to j in G with this property, and let j^* be the next-closer node on some shortest path from j'' to j in G. That is, $j'' \in \arg\min\{d_G(j,i') \mid \mathcal{C}_{i'}(t^*) \neq \mathcal{C}_j(t^*)\}$, $(j'', j^*) \in E$ and $d_G(j,j^*) = d_G(j,j'') - 1$. Recall that, in this case, there exists such a neighbor $j'' \in N^r_G(j)$ with this property, hence $0 \leq d_G(j,j^*) < d_G(j,j'') \leq r$.

To apply Lemma 3.11, with $z = d_G(j,j^*) \leq r-1$ and $t = t^*$, we first need to show that the three hypothesis of the lemma hold. First, by the second conjunct of (3), we have $|\mathcal{C}_j(t^*)| = \ell - 1$, which implies the first hypothesis of Lemma 3.11. Second, we need to show that $\mathcal{C}_j(t^*) \subseteq \mathcal{C}_{j'}(t^*)$, for every $j' \in N^z_G(j)$. This follows from the fact that $d_G(j,j'') = z+1$ and the fact that j'' is a closest node to j in G with the property that $\mathcal{C}_{j''}(t^*) \neq \mathcal{C}_j(t^*)$. This implies that $\mathcal{C}_j(t^*) = \mathcal{C}_{j'}(t^*)$, and in particular $\mathcal{C}_j(t^*) \subseteq \mathcal{C}_{j'}(t^*)$, for every $j' \in N^z_G(j)$, as needed. Third, we need to show that $|\mathcal{R}_{j'}(t^*)| \geq \ell$ for some neighbor $j' \in N^z_G(j)$. We show that $|\mathcal{R}_{j^*}(t^*)| \geq \ell$ (that is, j^*, in particular, does satisfy this property).

The fact that j'' is a closest node with this property and node j^* is closer than j'' to j in G, implies that $\mathcal{C}_{j^*}(t^*) = \mathcal{C}_j(t^*)$ and that $\mathcal{C}_{j''}(t^*) \neq \mathcal{C}_{j^*}(t^*)$. By the inductive hypothesis for $\ell - 1$, we obtain that either $m \in \mathcal{C}_{j''}(t^*)$ or $|\mathcal{C}_{j''}(t^*)| \geq \ell - 1$; either way, by Inequality (3), there is some message $m' \in \mathcal{C}_{j''}(t^*) \setminus \mathcal{C}_{j^*}(t^*)$, which implies by Part 3 of Lemma 3.5 (with $t = t' = t^*$, $i' = j''$ and $i = j^*$), that $m' \in \mathcal{R}_{j''}(t^*) \setminus \mathcal{C}_{j''}(t^*)$. This, in turn implies that $|\mathcal{R}_{j^*}(t^*)| \geq \ell$. Then Lemma 3.11 (using $z = d_G(j,j^*) \leq r-1$ and $t = t^*$), yields that $|\mathcal{R}_j(t^* + (r-1)F_{ack})| \geq \ell$. Since $t^* = t_{d+r,\ell-1} + F_{ack}$, by Part 2 of Lemma 3.12, we have that $t_{d,\ell} \geq t^* + (r-1)F_{ack}$. Thus, $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$ as needed for Part 1 of the lemma.

Case 2: $\mathcal{C}_{j'}(t^*) = \mathcal{C}_j(t^*)$, for all neighbors $j' \in N^r_G(j)$.

Since $G' \subseteq G^r$ [10], it holds, in particular, that $\mathcal{C}_{j'}(t^*) = \mathcal{C}_j(t^*)$, for all neighbors j' of j in G'. Let's focus on time $t_{d-1,\ell}$. By Part 3 and Part 4 of Lemma 3.12, we have

$$t^* \leq t_{d-1,\ell} + F_{prog} = t_{d,\ell}\ . \tag{4}$$

Now, if $|\mathcal{R}_j(t_{d-1,\ell})| \geq \ell$, then $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$, by Part 1 of Lemma 3.15 and we are done. So suppose that $|\mathcal{R}_j(t_{d-1,\ell})| \leq \ell - 1$. Recall that

$$|\mathcal{C}_j(t^*)| = \ell - 1 \text{ and } \mathcal{C}_j(t^*) \subseteq \mathcal{R}_j(t_{d-1,\ell}), \tag{5}$$

where the second inequality holds by combining Inequality (4) together with Part 2 of Lemma 3.5 (with $t = t^*$ and $t' = t_{d-1,\ell}$). This implies that $|\mathcal{R}_j(t_{d-1,\ell})| \geq \ell - 1$. Hence, $|\mathcal{R}_j(t_{d-1,\ell})| = \ell - 1$, which implies together with Inequality (5) that

$$\mathcal{R}_j(t_{d-1,\ell}) = \mathcal{C}_j(t^*). \tag{6}$$

Now we will apply Lemma 3.10, with $t = t_{d-1,\ell}$. To do this, we need to show the two hypotheses of that lemma: First, we need to show that $\mathcal{R}_j(t_{d-1,\ell}) \subseteq \mathcal{C}_{j'}(t_{d-1,\ell})$ for every neighbor j' of j in G'. Consider some neighbor j' of j in G^r. We have

$$\mathcal{R}_j(t_{d-1,\ell}) = \mathcal{C}_{j'}(t^*) \subseteq \mathcal{C}_{j'}(t_{d-1,\ell}), \text{ for every neighbor } j' \text{ of } j \text{ in } G^r,$$

[10] Note that this is the first place that we use this assumption.

where the first equality holds by the case analysis assumption and Equality (6); and the second inequality holds by combining the first inequality of (4) with Part 1 of Lemma 3.5. This implies, in particular, that $\mathcal{R}_j(t_{d-1,\ell}) \subseteq \mathcal{C}_{j'}(t_{d-1,\ell})$, for every neighbor j' of j in G' (since $G' \subseteq G^r$), as needed for the first hypothesis of Lemma 3.10.

To show the second hypothesis of Lemma 3.10, we need to show that $\mathcal{R}_{j'}(t_{d-1,\ell}) - \mathcal{R}_j(t_{d-1,\ell}) \neq \emptyset$, for some neighbor j' of j in G. So, fix a neighbor j^* of j in G at distance $d-1$ from i_0. By the inductive hypothesis for d, we obtain that either $m \in \mathcal{R}_{j^*}(t_{d-1,\ell})$ or $|\mathcal{R}_{j^*}(t_{d-1,\ell})| \geq \ell$; either way, by Inequality (3), there is some message $m' \in \mathcal{R}_{j^*}(t_{d-1,\ell}) \setminus \mathcal{C}_j(t^*)$.

Then, Lemma 3.10 yields that $|\mathcal{R}_j(t_{d-1,\ell} + F_{prog})| > |\mathcal{R}_j(t_{d-1,\ell})| = \ell - 1$. This implies that $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$, since $t_{d,\ell} = t_{d-1,\ell} + F_{prog}$ (by Part 4 of Lemma 3.12). Part 1 of the lemma follows.

2. Now, we prove Part 2 of the lemma. Before proceeding recall that $t^* = t_{d+r,\ell-1} + F_{ack} \leq t_{d,\ell} < t_{d,\ell} + F_{ack}$ (where the left inequality holds by Part 3 of Lemma 3.12), which implies together with Part 1 of Lemma 3.5, that

$$\mathcal{C}_j(t^*) \subseteq \mathcal{C}_j(t_{d,\ell}) \subseteq \mathcal{C}_j(t_{d,\ell} + F_{ack}).$$

Now, suppose that $m \in \mathcal{C}_j(t^*)$. Then, $m \in \mathcal{C}_j(t_{d,\ell} + F_{ack})$ (since $\mathcal{C}_j(t^*) \subseteq \mathcal{C}_j(t_{d,\ell} + F_{ack})$) and we are done. Next, assume that $m \notin \mathcal{C}_j(t^*)$. Then, by the inductive hypothesis for $\ell - 1$, $|\mathcal{C}_j(t^*)| \geq \ell - 1$. This implies, in particular, that $|\mathcal{C}_j(t_{d,\ell})| \geq \ell - 1$ (since $\mathcal{C}_j(t^*) \subseteq \mathcal{C}_j(t_{d,\ell})$). By Part 1, either $m \in \mathcal{R}_j(t_{d,\ell})$ or $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$; either way, we obtain that $|\mathcal{R}_j(t_{d,\ell})| \geq \ell$, since, $\mathcal{C}_j(t^*) \subseteq \mathcal{R}_j(t_{d,\ell})$, $|\mathcal{C}_j(t^*)| \geq \ell - 1$ and $m \notin \mathcal{C}_j(t^*)$. Then Corollary 3.8 implies that $|\mathcal{C}_j(t_{d,\ell} + F_{ack})| > |\mathcal{C}_j(t_{d,\ell})|$, so $|\mathcal{C}_j(t_{d,\ell} + F_{ack})| \geq \ell$ as needed.

□

3.2.5 The Main Theorem

Let $\mathcal{K} \subseteq \mathcal{M}$ be the set of messages that arrive at the nodes in a given execution α.

Theorem 3.16. *If $|\mathcal{K}| \leq k$ then $\mathcal{R}_j(t_1) = \mathcal{K}$ for every node j, where $t_1 = (D + (r+1)k - 2)F_{prog} + r(k-1)F_{ack}$.*

The conclusion of this theorem says all the messages of $\mathcal{K}$ are received at all nodes by time t_1.

Proof. Follows directly from Lemma 3.15. □

3.3 Lower Bound for Grey Zone G'

We are left with two questions concerning MMB. First, does BMMB perform as well as the r-restricted case when we we consider other natural restrictions on G', such as the grey zone constraint? And second, if not BMMB, are there *any* MMB algorithms for the standard abstract MAC layer model model that can perform well given the grey zone restriction, or perhaps even perform well for arbitrary G'? Below, we answer both questions in the negative by proving that all MMB algorithms require $\Omega((D + k)F_{ack})$ time to solve MMB with a grey zone restricted G'. This result establishes that our analysis of BMMB from Section 3.1 is tight, and it opens an intriguing gap between the superficially similar r-restricted and grey zone constraints.

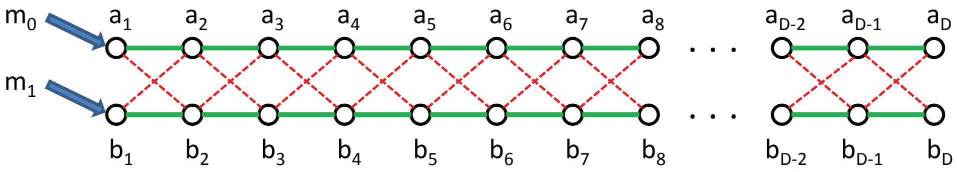

Figure 2: The lower bound network. The green solid lines represent the reliable edges, i.e., those in G, and the red dashed lines represent the unreliable edges, i.e., those in $G' \setminus G$.

Theorem 3.17. *For any Multi-Message Broadcast algorithm $\mathcal{A}$, every $k > 1$, and random bit assignment, there exists a network, message assignment, and message scheduler such that $\mathcal{A}$ requires $\Omega((D+k)F_{ack})$ time to solve the MMB problem.*

To prove our main theorem, we handle the kF_{ack} and DF_{ack} terms separately. The kF_{ack} part is simple: consider a node u that represents the only bridge in G between a receiver v and the source(s) of k messages. Our message size limit restricts u to send only a constant number of messages to v at a time, inducing the $\Omega(kF_{ack})$ bound.

Lemma 3.18. *For every $k \in [1, n-2]$, algorithm $\mathcal{A}$, and random bit assignment, there exists a network with $G' = G$, a message assignment that has no node begins with more than one message—what we call a* singleton assignment*—and a message scheduler such that $\mathcal{A}$ requires $\Omega(kF_{ack})$ time to solve the MMB problem.*

Proof. We first fix our definition of $G' = G$. To define G, connect each node in $U = \{u_1, u_2, ..., u_{k-1}\}$ to u_k, forming a star. Then connect u_k to some other node v. Start each node in $U \cup \{u_k\}$ with a unique broadcast message. Consider a message schedule that requires the full F_{ack} time between each broadcast and its corresponding acknowledgment. We now bound the time for v to receive all k messages. The key observation is that u_k is a choke-point through which all messages must pass to arrive at v. To bound the time for messages to make it through this constriction, divide time into *rounds* of length F_{ack}. By our assumption on the scheduler, u_k can begin the transmission of at most a constant number of message per round. Therefore, v can receive at most a constant number of new messages per round. The $\Omega(kF_{ack})$ bound follows directly. □

The more interesting step is proving the $\Omega(DF_{ack})$ term. To accomplish this goal, we begin by defining the network used in our proof argument. Fix some diameter D that is divisible by 2 (the below proof can be easily modified to handle odd D). Fix two node sets $U_A = \{a_1, a_2, ..., a_D\}$ and $U_B = \{b_1, b_2, ..., b_D\}$. Let A and B be the two line graphs that connect, in order of their indices, the nodes in U_A and U_B, respectively. Let C be the dual graph network over nodes $U_A \cup U_B$ where: G consists of the edges in A, B, and G' is defined to include all the edges in G, as well as the following extra edges: for $i < D$, a_i (resp. b_i) is connected to b_{i+1} (resp. a_{i+1}). Notice that our definition of G' in C satisfies the definition of grey zone restricted for a sufficiently large value for the constant c (see Section 2). An example of this network is shown in Figure 2.

In the following, we define a *endpoint-oriented* execution to be an execution of an MMB broadcast algorithm for $k = 2$, in C, for message set $M = \{m_0, m_1\}$, where m_0 starts at a_1 and m_1 starts at b_1. Given a finite endpoint-oriented execution α, let $\ell_0(\alpha)$ be the largest node in A (by increasing index order) that has received m_0 in α, and $\ell_1(\alpha)$ be the largest node in B that has received m_1. Let q_0 and q_1 be defined in the same way, except now capturing the largest node in the relevant line to have received *and initiated a broadcast* of the relevant message. Finally, we call an execution (or execution prefix/extension) *valid*, if the message events and their timing satisfy the model guarantees.

The main insight in our proof argument is that given the right scheduling strategy, m_0's progress down A can slow m_1's progress down B, as well as the other way around. This strategy, however, requires that nodes that receive these messages proceed to then broadcast them as well. The below lemma argues that when m_0 and m_1 make progress, either the next hops start broadcasting, or we can show that at least one of the messages is delayed long enough to establish our desired result.

Lemma 3.19. *Let α be a finite endpoint-oriented execution of a MMB algorithm $\mathcal{A}$ with random bit assignment κ in network C, such that $q_0(\alpha) < \ell_0(\alpha) = a_i$ and $q_1(\alpha) < \ell_1(\alpha) = b_i$ for some $i \in \{1, ..., D-1\}$. There exists a message schedule that produces one of the following outcomes: (1) an extension α' of α in which no time passes and $q_0(\alpha') = \ell_0(\alpha') = a_i$ and $q_1(\alpha') = \ell_1(\alpha') = b_i$; or (2) an extension α'' of α of duration $\Omega(DF_{ack})$ in which the MMB problem is not yet solved.*

Proof. Let α be the finite execution specified by the lemma statement. By assumption a_i has m_0 and b_i has m_1 at the end of α, but neither node has yet initiated a broadcast of this message. We begin by extending α using the following message schedule behavior:

For every broadcast initiated by a node in $U_A \cup U_B \setminus \{a_i, b_i\}$, or broadcast by a_i (resp. b_i) but not containing m_0 (resp. m_1): deliver the message to the broadcaster's neighbors in G (but to *no* G'-only neighbors) and then return an acknowledgment to the broadcaster, instantaneously (i.e., with no time passing). In scheduling these events, construct the schedule to proceed in a round robin fashion through all nodes; that is, for each node in this order, if there is a receive or acknowledgment event to schedule (as specified in the above rule), schedule that event and allow the node to initiate its next broadcast (if its algorithm dictates), then move on to the next node in the round robin order.

Call this extension β. Notice, β is not necessarily a valid execution of our algorithm because if a_i or b_i initiate a broadcast of m_0 and m_1, respectively, in β, they are starved by the schedule. We now use β, however, to force our algorithm to satisfy one of the two lemma outcomes. In more detail, let s_a be the step in β where a_i initiates a broadcast containing m_0 (define $s_a = \perp$ if no such step exists). Define s_b the same with respect to b_i and m_1. Because our schedule in β never allows these broadcasts to complete, there can only be at most one such step for a_i and b_i in β. We consider three cases depending on the values of s_a and s_b.

Case 1*: Assume that $s_a \neq \perp$ and $s_b \neq \perp$.* Let α' be the prefix of β that stops at whichever of these two steps happens later in β. Notice, α' provides a valid extension of α: even though either a_i or b_i might have been delayed from delivering a message in this extension, no time passed between α and the end of α', so no timing guarantees were violated. Accordingly, we see that α' satisfies outcome (1) of the lemma statement.

Case 2*: $s_a = s_b = \perp$.* In this case, β does not starve any node: every initiated broadcast is delivered to G neighbors and acknowledged. Let α' be a transformation of β where we: (1) allow F_{prog} time to pass between each broadcast and its corresponding acknowledgment; and (2) we stop after DF_{ack} time has passed since the end of α. Because our algorithms are event-driven (and therefore have no concept of time), it is straightforward to see that α' is indistinguishable from β for all nodes. We also node that the schedule in α' satisfies the necessary time constraints, as we never delay a pending delivery by more than F_{prog} time. It follows that α satisfies outcome (2) of the lemma statement.

Case 3: either $s_a = \perp$ or $s_b = \perp$, but not both. Assume, w.l.o.g., that $s_a = \perp$ (the other case is symmetric). Let β' be an extension of α defined with the same rules as β with two exceptions: (1) schedule b_i's broadcasts of m_1 the same as all other broadcasts; and (2) allow F_{prog} time to pass between each broadcast and its corresponding acknowledgment. Let α' be the prefix of β' that ends after DF_{ack} time has passed since the end of α. As in the previous case, we note that α' is indistinguishable from β with respect to nodes in A (the scheduling rules defined above for β do not allow messages from B to be delivered to nodes in A, therefore nodes in A cannot learn that, in β', b_i can succeed in its broadcasts of m_1) and that it still satisfies the model's time bounds. As a result, a_i behaves the same in α' as in β and does not broadcast m_0. Because a_i, by assumption, is the furthest node down the line in A to receive m_0 so far, it follows that by the end of α' there are nodes in A that have not yet received m_0. It follows that α' satisfies outcome (2). □

With the above lemma established, we can use it to prove the main lemma regarding the necessity of DF_{ack} rounds. Here is the main idea: As the messages arrive at each new hop in their respective lines, we apply the above lemma to force these new hops to initiate broadcasts (or, directly prove our time bound by delaying too long). Once we have established that the message frontiers on each line are broadcasting, we can allow these broadcasts to mutually interfere over $G' \setminus G$ edges in such a way that satisfies the progress bound while preventing useful dissemination.

Lemma 3.20. *For every algorithm $\mathcal{A}$ and random bit assignment κ, there exists a message assignment and schedule such that $\mathcal{A}$ requires $\Omega(DF_{ack})$ time to solve MMB in network C for $k = 2$.*

Proof. We construct an endpoint oriented execution of $\mathcal{A}$ in C with random bits κ, by defining the message schedule behavior. We start with α_0: the finite execution that captures the behavior of the above system only through a_1 receiving m_0 and b_1 receiving m_1. These events happen at the beginning of the execution, so no time passes in α_0.

Notice, α_0 satisfies the preconditions required to apply Lemma 3.19. Apply this lemma to α_0. By the definition of this lemma, there are two possible outcomes. If it is the second outcome, we have proved our theorem. Assume, therefore, that the lemma produces an extension α_0' of α_0 that satisfies the first outcome. At the end of α_0', we know that a_1 has initiated a broadcast of m_0 and b_1 has initiated a broadcast of m_1, and no time has passed since these broadcasts are initiated. We further note that at this point, m_0 has made it no further down the A line and b_1 has made it no further down the B line.

We now extend α_0' with a message schedule that delays m_0's arrival at a_2 and m_1's arrival at b_2 by the maximum F_{ack} time. To do so, partition an interval of F_{ack} time following α_0' into sub-intervals of length F_{prog}. At the end of each sub-interval, deliver m_0 from a_1 to b_2 (over a G' edge) and m_1 from b_1 to a_2 (also over a G' edge). At the end of this F_{ack} interval, allow m_0 to make it to a_2 and m_1 to make it to b_2, and acknowledge these broadcasts. Notice, this schedule satisfies both the progress and acknowledgment bounds for a_1 and b_1's broadcasts during this interval. During this F_{ack} interval, however, we must also schedule *other nodes'* broadcasts. To do so, we use a simple rule: for every other broadcast, allow the message to be delivered to all (and only) G neighbors and be acknowledged at the end of the next F_{prog} interval.

Notice, our above delay strategy leads to a finite execution α_1, of duration F_{ack} longer than α_0, where $q_0(\alpha_1) \neq \ell_0(\alpha_1) = a_2$ and $q_1(\alpha_1) \neq \ell_1(\alpha_1) = b_2$. We can, therefore, apply our above argument again, now replacing α_0 with α_1. Indeed, we can keep applying this argument until either we arrive at outcome (2) from Lemma 3.19, or we build up to α_{D-2}, an execution of length

$\Omega(DF_{ack})$ in which m_0 and m_1 have not yet made it to the end of the A and B lines, respectively. Either way, we have proved the theorem statement. □

4 Multi-Message Broadcast with an Enhanced Abstract MAC Layer

In Section 3, we proved that in the standard abstract MAC layer model, $\Omega(kF_{ack})$ time is necessary to solve MMB, and for some definitions of G', an additional $\Omega(DF_{ack})$ time is also necessary. Our analysis of BMMB then established that this algorithm is essentially the best you can do in this model. In this section, we tackle the question of how much *additional power* we must add to our model definition to enable faster solutions under the assumption that $F_{prog} \ll F_{ack}$, pointing to the extra assumptions of the enhanced abstract MAC layer model as one possible answer. In particular, we describe a new algorithm, which we call *Fast Multi-Message Broadcast* (FMMB), that guarantees the following time complexity when run in the enhanced abstract MAC layer model with a grey zone restricted G':

Theorem 4.1. *The FMMB algorithm solves the MMB problem in $O((D\log n + k\log n + \log^3 n)F_{prog})$ time, w.h.p., in the enhanced abstract MAC layer model and grey zone restricted G'.*

This result has no F_{ack} term. As the size of F_{prog} decreases, this result's advantage over BMMB increases.

Preliminaries. In the following, for $v \in V$, we use $ID(v)$ to refer to v's unique id, $N_G(v)$ to describe *the ids* of v's neighbors in G, and $N_{G'}(v)$ to describe the ids of v's neighbors in G'. We use $\mathcal{M}$ to refer to the set of messages to be disseminated in a given execution of MMB. We call a set $S \subseteq V$ of nodes *G-independent* if for each pair of nodes $u, v \in S$, we have $(v, u) \notin E$. In our analysis, we make frequent use of the following well-known fact, sometimes referred to as the *Sphere Packing* Lemma.[11]

Lemma 4.2. *Consider $P \subseteq \mathbb{R}^2$ such that $\forall p_1 \neq p_2 \in P$, we have $1 < \|p_1 - p_2\|_2 \leq d$. Then $|P| = O(d^2)$.*

4.1 Algorithm Outline

The FMMB algorithm divides time into lock-step *rounds* each of length F_{prog}. This can be achieved by leveraging the ability of a node to use time and *abort* a broadcast in progress in the enhanced abstract MAC layer. In more detail, when we say a node *broadcasts in round t*, we mean that it initiates the broadcast at the beginning of the time slot dedicated to round t, and aborts it (if not completed yet) at the end of the time slot.

The FMMB algorithm uses three key subroutines which we summarize here, but detail and analyze in the subsections that follow. All three subroutines are randomized and will be shown to hold with sufficiently high probability that their correctness guarantees can be combined with a union bound. The FMMB algorithms begins by having nodes construct a maximal independent set (MIS) in G using $O(\log^3 n)$ rounds. We note that this MIS subroutine might be of independent interest.[12] The FMMB algorithm then uses a gather subroutine to gather the broadcast messages

[11] Although the precise constants in the bound on the cardinality of S in Lemma 4.2 are known, the asymptotic version stated above is sufficient for our purposes.

[12] The previously best known MIS solution for an abstract MAC layer model uses time that is linear in n [32].

at nearby MIS nodes in an additional $O(k + \log n)$ rounds. Finally, it uses an overlay dissemination subroutine that broadcasts the messages to all MIS nodes, and then to their neighbors (i.e., all nodes), in $O((D + k) \log n)$ rounds. The total combined running time of FMMB is therefore $O(D \log n + k \log n + \log^3 n)$ rounds, which requires $O((D \log n + k \log n + \log^3 n)F_{prog})$ total time.

We continue by explaining each of the three subroutines. Theorem 4.1 follows directly Lemmata 4.5, 4.6, and 4.8. We also note that all three subroutines depend on the assumption of a grey zone restricted G', which is leveraged in our analysis to enforce useful regionalization properties on the MIS nodes.

4.2 The MIS Subroutine

We now describe an MIS subroutine that succeeds in building an MIS in G in $O(c^4 \log^3 n)$ rounds, w.h.p., where c is the universal constant from the grey zone definition (see Section 2). In more detail, the algorithm runs for a fixed length of time, $t_{MIS} \in O(c^4 \log^3 n)$. At the end of this period, some set $S \subseteq V$ of nodes *join* the MIS. The algorithm guarantees, w.h.p.,[13] that S is a *maximal G-independent set*: (1) all pairs of nodes in S are G-independent; and (2) every $u \in V$ is either in S or neighbors a node in S in G.

The subroutine (called "algorithm" from here forward) works as follows: initially, all nodes are active. In the course of the algorithm, some nodes join the MIS and some nodes become inactive. The algorithm runs in $O(c^2 \log^2 n)$ phases, each of which consists of $O(c^2 \log n)$ rounds, which are divided into two parts: *election* and *announcement*.

The election part has $4 \log n$ rounds. At the start, each active node v picks a random bit-string $b(v) \in \{0, 1\}^{4 \log n}$. In each round $\tau \in [1, 4 \log n]$ of this part, each active node v broadcasts its bit-string $b(v)$ iff the τ^{th} bit of $b(v)$ is 1. If node v did not broadcast but it received a message $b(u)$, be it from a G or a G' neighbor, then node v becomes temporarily inactive for the rest of this phase. At the end of $4 \log n$ rounds of the election part, if a node v is still active, then v joins the MIS set S.

The announcement part has $O(c^2 \log n)$ rounds. In each round, each node v that joined the MIS in this phase broadcasts a message containing $ID(v)$ with probability $\Theta(1/c^2)$, and does not broadcast any message with probability $1 - \Theta(1/c^2)$. If a node u that has not joined the MIS receives a message $ID(v)$ from a G-neighbor, then u knows that one of its G-neighbors is in the MIS and thus node u becomes permanently inactive. At the end of the announcement part, each node that joined the MIS in this phase becomes permanently inactive, while each temporarily inactive node becomes active again.

Lemma 4.3. *The set S of nodes that join the MIS nodes is G-independent with high probability.*

Proof. We show that the probability that there are two G-neighbors $v, u \in S$ is at most $\frac{1}{n}$.

First suppose that there are two G-neighbors v and u that joined the MIS in the same phase. Then, it must hold that in that phase $b(v) = b(u)$. This is because, otherwise there would exist a round of the election part where one of the two nodes, say v, is not broadcasting but the other one, u, is broadcasting. In that case, v would receive the message of a G'-neighbor—which might be v or not—and thus become temporarily inactive which means that v would not join the MIS in this phase. It is an easy calculation to see that the probability that $b(v) = b(u)$ is at most $\frac{1}{n^4}$ and

[13] For this guarantee, as with the other subroutines we consider, we assume that the high probability is of the form $1 - n^{-c}$ for a sufficiently large constant c to enable a union bound on the fail probability for all three subroutines.

a union bound over all choices of the pair u, v establishes that the probability of existence of such a pair is at most $\frac{1}{n^2}$.

Now suppose that there were not two G-neighbors that joined the MIS in the same phase but there were two nodes that joined the MIS in different phases. Let t be the first phase in which there are two G-neighbors v, u that are in the MIS. Without loss of generality, suppose that v was not in the MIS at the end of phase t and u joined the MIS in phase $t' < t$. The set S' of G'-neighbors of v that joined the MIS in phase t' is a G-independent set. Hence, using Lemma 4.2, we get that $|S'| = O(c^2)$. Now in each round of the announcement part of t', node u broadcasts with probability $\Theta(1/c^2)$ and each other node in S' does not broadcast with probability $\Theta(1/c^2)$. Hence, the probability that v receives the message of u in one round is at least $\Theta(1/c^2)(1-\Theta(1/c^2))^{|S'|-1} \geq \Theta(1/c^2)(1-\Theta(1/c^2))^{O(c^2)} = \Theta(1/c^2)$. Hence, during the $\Theta(c^2 \log n)$ rounds of the announcement part, v receives the message of u with probability at least $1 - \frac{1}{n^4}$. Hence, the probability does not receive this message and later joins the MIS is at most $\frac{1}{n^4}$. Again, taking a union over all node pairs shows that the probability of existence of such a pair u, v is at most $\frac{1}{n^2}$.

Overall, using another union bound over the two cases considered in the above two paragraphs, we get that the probability that there are two G-neighbors $v, u \in S$ is at most $\frac{1}{n}$. □

Lemma 4.4. *For each phase t and each node v. If at the start of phase t, node v is active, then in phase t, at least one node u such that $\|p(v) - p(u)\|_2 = O(c \log n)$ joins the MIS.*

Proof. Consider a phase t and a node v that is active at the start of phase t. We use a node variable u_τ, for round $\tau \in [1, 4 \log n]$ of the election part, to keep track of a node that is still active. Initially, $u_1 = v$. For each round τ, if in round τ, node u_τ broadcasts or it does not broadcast but it also does not receive a message, then let $u_{\tau+1} = u_\tau$. If $u\tau$ does not broadcast in round c but it receives a message from a G'-neighbor w, then let $u_{\tau+1} = w$. It follows from this recursive definition that $u_{4\log n}$ is active at the end of round $4 \log n$ of the election part and hence, $u = u_{4 \log n}$ joins the MIS. Furthermore, it is easy to see that $\|p(u_{\tau+1}) - p(u_\tau)\|_2 \leq \tau$ and hence, using the triangular inequality, we have $\|p(u) - p(v)\|_2 = \|p(u_{4\log n}) - p(u_1)\| \leq \sum_{\tau=1}^{4\log n - 1} \|p(u_{\tau+1}) - p(u_\tau)\|_2 \leq 4c \log n$. This completes the proof. □

Lemma 4.5. *The set S of nodes that join the MIS is a maximal G-independent set with high probability.*

Proof. The proof requires us to establish two properties: (A) w.h.p., no two nodes $v, u \in S$ are G-neighbors, and (B) w.h.p., each node $v \in V \setminus S$ has a G-neighbor in S. Property (A) follows directly from Lemma 4.3. We now prove property (B). Consider a node $v \in V$, suppose that v does not join the MIS, and let S' be the set of nodes within distance $O(c \log n)$ of v that join the MIS. Using Lemma 4.4, we know that for each phase t in which v starts as an active node, at least one new node joins S'. On the other hand, from Lemma 4.3, we know that the set of nodes that join MIS and thus also S' is w.h.p. a G-independent set. Hence, using Lemma 4.2, we get that $|S'| = O(c^2 \log^2 n)$. It follows that node v cannot be active at the start of more than $O(c^2 \log^2 n)$ phases, which means that there is a phase in which v becomes permanently inactive. Recalling the description of the algorithm, we get that this means that in the announcement part of that phase, v receives the message of a G-neighbor that has just joined the MIS. Hence, v indeed has a G-neighbor in the MIS set S, which proves property B. □

4.3 The Message Gathering Subroutine

We now describe a message gathering subroutine (called "algorithm" in the rest of this subsection) that delivers each MMB message to a nearby MIS node in $O(c^2(k + \log n))$ rounds, w.h.p. In more detail, each node v maintains message-set $M_v \subseteq \mathcal{M}$ of messages that the node currently owns. When this algorithm is first called, these sets describe the initial assignment of MMB message to nodes. Throughout the algorithm, the message-set of MIS nodes grow while the message set of non-MIS nodes shrink. The goal is to arrive at a configuration where $\cup_{v \in S} M_v = \mathcal{M}$: at which point, all messages in $\mathcal{M}$ are owned by MIS nodes. The algorithm is divided into $O(c^2(k + \log n))$ *periods*, where each period consists of three rounds. At the start of each period, each MIS node decides to be active with probability $1/\Theta(c^2)$, and inactive otherwise. Then, in the first round of the period, each active MIS node broadcasts its ID, announcing that it is active. In the second round, each non-MIS node v that received a message from one of its G-neighbors in the first round and has at least one message left in its message-set M_v broadcasts one of the messages in M_v, along with its own ID. In the same round, if an MIS node u receives a message m from a G-neighbor, then node u updates its message-set as $M_u = M_u \cup \{m\}$. In the third round of the period, each MIS node u that received a message m in the second round sends an *acknowledgment* message, which contains message m and its own ID. In this round, if a non-MIS node v receives a message m from a G-neighbor, then v updates its message-set as $M_v = M_v \setminus \{m\}$.

Lemma 4.6. *When the above algorithm is executed given a valid MIS S, the following holds at termination, w.h.p.: $\cup_{v \in S} M_v = \mathcal{M}$. That is, each message is owned by at least one MIS node.*

Proof. Consider a non-MIS node v and suppose that at the start of the algorithm, node v has message-set $M_v = T_0 \neq \emptyset$. We show that at the end of the algorithm, w.h.p., each message $m \in T_0$ is held by at least one MIS node u. Fix u to be one (arbitrary) G-neighbor of v that is in the MIS set. Let $A_u \subseteq \mathcal{M}$ be the set of messages for which u has broadcast an acknowledgment and this acknowledgment is received by all G-neighbors of u. We prove that in each period in which $M_v \neq \emptyset$, with probability at least $1/\Theta(c^2)$, $|A_u|$ increases by one.

Let S_u be the set of all MIS nodes that are within distance $2c$ of u. Using Lemma 4.2, we know that $|S_u| = O(c^2)$. Therefore, for each period t, the probability that u is the only MIS node in S_u that is active in period t is at least $1/\Theta(c^2)(1-1/\Theta(c^2))^{O(c^2)} = 1/\Theta(c^2)$. Suppose that u is the only MIS node in S_u that is active in period t. Furthermore, assume that $M_v \neq \emptyset$. Then, in the second round of period t, the only G'-neighbors of u that are broadcasting are in fact G-neighbors of u. This is because, consider a node w that is a G'-neighbor of u but not a G-neighbor of u and suppose that w is broadcasting in the second round of period t. Then an MIS G-neighbor $w' \neq u$ of w must be active in this period. It follows that $\|p(w') - p(u)\|_2 \leq \|p(w') - p(w)\|_2 + \|p(w) - p(u)\|_2 \leq 1 + c$. Thus, $w' \in S_u$ which is in contradiction with the assumption that u is the only active node in S_u. Now, in period t, node v broadcasts a message in M_v. Note that by the description of the algorithm $M_v \cap A_u = \emptyset$. Hence, we conclude that u receives a message m from one of its G-neighbors and this message is not in A_u. In the third round of this period, u acknowledges this message m. We claim that this acknowledgment is received by all G-neighbors of u, which means that $|A_u|$ increases by one. The reason is that, if a G-neighbor w of u does not receive the acknowledgment, it means that a G'-neighbor $w' \neq u$ of w was broadcasting in the third round. By the description of the algorithm, we get that w' is an active MIS node, and furthermore, $\|p(w') - p(u)\|_2 \leq \|p(w') - p(w)\|_2 + \|p(w) - p(u)\|_2 \leq c + 1$, which means that $w' \in S_u$, which is a contradiction to the assumption that u is the only active MIS node in S_u in period t. Hence, we have established that in each period in which $M_v \neq \emptyset$, with probability at least $1/\Theta(c^2)$, $|A_u|$ increases by one. Hence, in expectation, after $O(kc^2)$ such periods, $|A_u| \geq k$. That is, the set M_v is emptied which means that for each message m that was originally in M_v, v has received an acknowledgment and thus, the message m is now held by at least one MIS nodes. A basic application of Chernoff bound then shows that after $O(c^2(k + \log n)) = O(c^2(k + \log n))$, w.h.p. we have $|A_u| \geq k$ and thus, each message m intially held by v is now held by at least one MIS nodes. Taking a union bound over all non-MIS nodes v then completes the proof. □

4.4 The Message Spreading Subroutine

We conclude by describing the subroutine ("algorithm" in the following subsection) used by FMMB to efficiently spread the messages gathered at MIS nodes to the full network. This algorithm spreads the messages to all nodes in the network in $O((D + k) \log n)$ rounds, w.h.p. In more detail, in the following, let S be the set of MIS nodes when this algorithm is executed. Assume S is a valid MIS. Let E_S be the set of unordered pairs $(v, u) \in E$ such that the hop distance of u and v in graph G is at most 3. Consider the overlay graph $\mathcal{H} = (S, E_S)$. The algorithm works by spreading messages over $\mathcal{H}$. For this purpose, we explain a simple procedure, that uses $O(\log n)$ rounds, and that achieves the following: Suppose that each node $v \in S$ starts this procedure with at most one message m_v. Then, at the end of this procedure, w.h.p., we have that m_v is delivered to all $\mathcal{H}$-neighbors of v. We will then establish the final upper bound of $O((D + k) \log n)$ rounds by combining this procedure with a standard pipelining argument applied to messages in $\mathcal{H}$.

The Local Broadcast Procedure on the Overlay. The algorithm consists of $O(c^2 \log n)$ periods, each consisting of three rounds. In each period, each node v decides to be active with probability $1/\Theta(c^2)$ and remains inactive otherwise. If a node $v \in S$ is active, it broadcasts its message m_v in the first round, if it has a message m_v. For all the three rounds of the period, if a node $u \in V$ receives a message from a G-neighbor in one round, it broadcasts this message in the next round. At the end of the three rounds of the period, each node $u \in S$ adds the messages that it has received to its message-set.

Lemma 4.7. *At the end of the procedure, we have that for each node $v \in S$, if v starts the procedure with message m_v, then m_v is delivered to all $\mathcal{H}$-neighbors of v with high probability.*

Proof. Let S_v be the set of nodes $u \in S$ such that $\|p(v) - p(u)\|_2 \leq 7c$. For now suppose that v is the only node in S_v that is active. We claim that in this case, in the τ^{th} round of the period—where $\tau \in \{1, 2, 3\}$, all nodes that their G-distance to v is τ hops receive m_v. Hence, overall the three rounds, all $\mathcal{H}$-neighbors of v receive m_v. The proof of this claim is as follows. First consider the case $\tau = 1$. Then, if there is a G-neighbor w of v such that w does not receive m_v in the first round, it would mean that w has a G'-neighbor w' that is in S and is active in this period. We have $\|p(w') - p(v)\|_2 \leq \|p(w') - p(w)\|_2 + \|p(w) - p(v)\|_2 \leq c + 1$. Thus, w is in S_v which is in contradiction with the assumption that v is the only node in S_v that is active. Now we move to proving the claim for $\tau = 2$ or $\tau = 3$. Suppose that τ^* is the smallest $\tau \in \{2, 3\}$ for which the claim breaks and there is a node w that has G-distance of τ from v but it does not receive m_v in round τ. We know that w has a G-neighbor w' that has G-distance $\tau^* - 1$ from v and w' receives m_v in round $\tau^* - 1$. Hence, there must be a G'-neighbor w'' of w that broadcasts a message $m' \neq m_v$ in round τ^*. Given the description of the algorithm, it follows that there is an active node $u \in S$ which started message m' in this period and u is has G-distance at most τ^* from w. Thus, we get

$\|p(v) - p(u)\|_2 \leq \|p(v) - p(w'')\|_2 + \|p(w'') - p(w)\|_2 + \|p(w) - p(v)\|_2 \leq \tau^* + c + \tau^* \leq c + 6 \leq 7c$. This means that w'' is in S_v which is in contradiction with the assumption that v is the only node in S_v that is active. This contradiction completes the proof of the claim, establishing that if v is the only node in S_v that is active, then m_v is delivered to all $\mathcal{H}$-neighbors of v. Now note that using Lemma 4.2, we get $|S_v| = O(c^2)$. Thus, in each period, the probability that v is the only node in S_v that is active is $1/\Theta(c^2)(1 - 1/\Theta(c^2))^{O(c^2)} = 1/\Theta(c^2)$. Hence, in $O(c^2 \log n)$ periods, with high probability, there is at least one period in which v is the only node in S_v that is active. Therefore, with high probability, m_v gets delivered to all $\mathcal{H}$-neighbors of v. Taking a union bound over all choices of node v completes the proof. □

This local broadcast on the overlay provides essentially the same guarantee as given by F_{ack} on the full network topology, but with respect to the overlay graph $\mathcal{H}$. Having this *simulated broadcast*, the problem can be solved by combining BMMB with this simulated broadcast, and then analyzing its performance with respect to $\mathcal{H}$. That is, we divide the time into phases, each of length $O(\log n)$ rounds, where the constants are such that one run of the above procedure fits in one phase. Then, in each phase, each MIS node sends a message that it has not sent so far, to all of its $\mathcal{H}$-neighbors. It follows from Theorem 3.1 that after $O(D_{\mathcal{H}} + k)$ phases, all messages are broadcast over $\mathcal{H}$, i.e., to all MIS nodes. Here $D_{\mathcal{H}}$ is the hop diameter of the overlay graph $\mathcal{H}$, and we clearly have $D_{\mathcal{H}} \leq D_G = D$. Below is a more detailed description of this part, as well as the final lemma statement for this subroutine.

Broadcast on the Overlay Graph $\mathcal{H}$. Here, we explain a more detailed version of the algorithm that broadcasts messages on the overlay graph $\mathcal{H}$, in $O((D + k) \log n)$ rounds. We divide the $O((D + k) \log n)$ rounds into $O(D + k)$ phases, each of length $O(\log n)$ rounds, where the constants are such that one run of the above procedure fits in one phase. In the algorithm, each node $v \in S$ has a message-set M_v of messages that it has or it has received, and it also has a *sent-set* M'_v of messages that contains all the messages that v has sent throughout this algorithm. Initially, for each node v, $M'_v = \emptyset$. In each phase, each node v sets m_v to be equal to one of the messages in $M_v \setminus M'_v$ and runs the procedure explained above. At the end of the phase, node v adds m_v to M'_v and it also adds each message received during this phase to M_v. The following theorem shows that this algorithm broadcasts all messages to all MIS nodes.

Lemma 4.8. *At the end of $D_{\mathcal{H}} + k$ phases, for each node $v \in S$, we have $M_v = \mathcal{M}$. Here $D_{\mathcal{H}}$ is the hop diameter of the overlay graph $\mathcal{H}$ and we clearly have $D_{\mathcal{H}} \leq D_G = D$ with high probability.*

Proof. Consider a message $m \in \mathcal{M}$ and let $S_m \subseteq S$ be the set of nodes $u \in S$ that hold m at the start of the algorithm. For each node $v \in S$, each $d \in [1, D_{\mathcal{H}}]$, and $\ell \in [1, k]$, set $t_{d_v,\ell}(v) = d_v + \ell$, where d_v is the $\mathcal{H}$-distance of node v to the set S_m, that is, the smallest d such that there is a node $u \in S_m$ that is within d $\mathcal{H}$-hops of v.

We claim that for each node v, after $t_{d,\ell}(v)$ phases, node v has m or ℓ other messages in its sent-set M'_v, w.h.p. It would then immediately follow that after $t_{D_{\mathcal{H}},k}(v)$, node v has m in its sent-set M'_v and hence also in M_v.

We prove the claim using an induction on $h = d_v + \ell$. The base case $h = 0$ is straightforward as when $h = 0$, we also have $d_v = 0$ and in that case, the claim reduces to a trivial statement about the local queue of node v: Namely that if node v has message m in its local queue at the start of the algorithm, then after ℓ phases, v has either message m or ℓ other messages in its sent-set M'_v. For the inductive step, consider a node $v \in S$ such that $d_v + \ell = h$. If $d_v = 0$, then the claim follows from the same trivial local-queue argument. Suppose that $d_v \geq 1$ and consider an $\mathcal{H}$-neighbor u of v such that $d_u = d_v - 1$. By the induction hypothesis, we know that by the end of phase $h - 1 = d_v + \ell - 1$, v has either m or at least $\ell - 1$ other messages in M'_v, and u has either m or at least ℓ other messages in M'_u. For each of these four possibilities, we get that with high probability, by the end of phase $h = d_v + \ell$, node v has either m or at least ℓ other messages in M'_v. This is because of the following: if v already has m or at least ℓ other messages in M'_v at the end of phase $h - 1 =$, then we are done. Otherwise, using Lemma 4.7, we get that w.h.p., by the end of phase $h - 1$, node v has received either m or ℓ other messages from u which shows that at the start of phase h, node v has at least one message in $M_v \setminus M'_v$, either m or a different message. Thus, at the end of phase h, either m or ℓ other messages are in M'_v. This finishes the proof. □

5 Conclusion

In this paper, we applied the abstract MAC layer approach to a natural problem: disseminating an unknown amount of information starting at unknown devices through an unknown network (what we call multi-message broadcast). We proved that the presence of unreliable links has a significant but perhaps unexpected impact on the worst-case performance of multi-message broadcast. In particular, with no unreliability or unreliable links limited to nodes close in the reliable link graph, basic flooding (what we called the BMMB algorithm) is efficient. Once we shift to the similar constraint of unreliability limited to nodes close in geographic distance, however, all solutions are inherently slow. This indicates an interesting property of unreliability: the ability to unreliably connect nodes distant in the reliable link graph seems to be what degrades worst-case performance of broadcast algorithms. Finally, we demonstrated that if nodes have estimates of the model time bounds and can abort messages in progress, even more efficient solutions to this problem are possible. Most existing MAC layers do not offer an interface to abort messages. This result motivates the implementation of this interface (which seems technically straightforward).

In terms of future work, there exist many other important problems for which a similar analysis can be performed, such as leader election, consensus, and network structuring. It would be interesting to investigate whether there are properties of link unreliability that are universal to distributed computation in this setting, or if the properties of this type that matter differ from problem to problem. Another direction to study within this same general area is whether the strength of the scheduler strongly impacts worst-case performance. In our lower bound, for example, the scheduler knows the algorithm's random bits. This is a strong assumption and motivates the question of whether this bound can be circumvented with a weaker adversary and a more clever algorithm.

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. On the Time Complexity of Broadcast in Radio Networks: an Exponential Gap Between Determinism and Randomization. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 1987.

[3] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. Syst. Sci.*, 45(1):104–126, 1992.

[4] K. Censor-Hillel, Seth Gilbert, N. Lynch, and Calvin Newport. Structuring Unreliable Radio Networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2011.

[5] I. Chlamtac and S. Kutten. On broadcasting in radio networks: Problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.

[6] Hyun Chul Chung, Peter Robinson, and Jennifer L. Welch. Regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*, pages 81–90, 2010.

[7] Hyun Chul Chung, Peter Robinson, and Jennifer L. Welch. Optimal regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of the 7th ACM ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing*, FOMC '11, pages 52–61, 2011.

[8] A. E. F. Clementi, A. Monti, and R. Silvestri. Round Robin is Optimal for Fault-Tolerant Broadcasting on Wireless Networks. *Journal of Parallel and Distributed Computing*, 64(1):89–96, 2004.

[9] Alejandro Cornejo, Nancy Lynch, Saira Viqar, and Jennifer L Welch. Neighbor Discovery in Mobile Ad Hoc Networks Using an Abstract MAC Layer. In *Annual Allerton Conference on Communication, Control, and Computing*, 2009.

[10] Alejandro Cornejo, Saira Viqar, and Jennifer L Welch. Reliable Neighbor Discovery for Mobile Ad Hoc Networks. In *Proceedings of the Workshop on the Foundations of Mobile Computing*, 2010.

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

[12] A. Czumaj and W. Rytter. Broadcasting Algorithms in Radio Networks with Unknown Topology. *Journal of Algorithms*, 60:115–143, 2006.

[13] Sebastian Daum, Seth Gilbert, Fabian Kuhn, and Calvin Newport. Broadcast in the Ad Hoc SINR Model. In *Proceedings of the International Symposium on Distributed Computing*, 2013.

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

[15] L. Gasieniec, D. Peleg, and Q. Xin. Faster communication in known topology radio networks. *Distributed Computing*, 19(4):289–300, 2007.

[16] Leszek Gasieniec, Andrzej Pelc, and David Peleg. The Wakeup Problem in Synchronous Broadcast Systems. *SIAM Journal on Discrete Mathematics*, 14(2):207–222, 2001.

[17] M. Ghaffari, B. Haeupler, and M. Khabbazian. Randomized broadcast in radio networks with collision detection. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2013.

[18] Mohsen Ghaffari, Bernhard Haeupler, Nancy Lynch, and Calvin Newport. Bounds on Contention Management in Radio Networks. In *Proceedings of the International Symposium on Distributed Computing*, 2012.

[19] Mohsen Ghaffari, Nancy Lynch, and Calvin Newport. The Cost of Radio Network Broadcast for Different Models of Unreliable Links. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2013.

[20] Tomasz Jurdzinski, Dariusz R. Kowalski, Michal Rozanski, and Grzegorz Stachowiak. Distributed Randomized Broadcasting in Wireless Networks under the SINR Model. In *Proceedings of the International Symposium on Distributed Computing*, 2013.

[21] Dilsun K Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. The theory of Timed I/O Automata. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–137, 2010.

[22] Thomas Kesselheim and Berthold Vöcking. Distributed Contention Resolution in Wireless Networks. In *Proceedings of the International Symposium on Distributed Computing*, 2010.

[23] M. Khabbazian and D. Kowalski. Time-efficient randomized multiple-message broadcast in radio networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 373–380, 2011.

[24] Majid Khabbazian, Fabian Kuhn, Dariusz Kowalski, and Nancy Lynch. Decomposing Broadcast Algorithms Using Abstract MAC Layers. In *Proceedings of the Workshop on the Foundations of Mobile Computing*, 2010.

[25] Majid Khabbazian, Fabian Kuhn, Nancy Lynch, Muriel Médard, and Ali ParandehGheibi. MAC Design for Analog Network Coding. In *Proceedings of the Workshop on the Foundations of Mobile Computing*, 2011.

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

[27] D. Kowalski and A. Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, 2007.

[28] D.R. Kowalski and A. Pelc. Broadcasting in Undirected Ad Hoc Radio Networks. *Distributed Computing*, 18(1):43–57, 2005.

[29] Fabian Kuhn, Nancy Lynch, and Calvin Newport. The Abstract MAC Layer. In *Proceedings of the International Symposium on Distributed Computing*, 2009.

[30] Fabian Kuhn, Nancy Lynch, and Calvin Newport. The Abstract MAC Layer. *Distributed Computing*, 24(3-4):187–206, 2011.

[31] E. Kushilevitz and Y. Mansour. An $O(Dlog(N/D))$ lower bound for broadcast in radio networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 65–74, 1993.

[32] Nancy Lynch, Tsvetomira Radeva, and Srikanth Sastry. Asynchronous leader election and mis using abstract mac layer. In *Proceedings of the 8th International Workshop on Foundations of Mobile Computing*, pages 3:1–3:10, 2012.

[33] Calvin Newport, David Kotz, Yougu Yuan, Robert S Gray, Jason Liu, and Chip Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *Simulation*, 83(9):643–661, 2007.

[34] A. Pelc. Algorithmic aspects of radio communication. pages 1–2, 2008.