Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2015-016 May 18, 2015

A (Truly) Local Broadcast Layer for Unreliable Radio Networks

Nancy Lynch and Calvin Newport

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

A (Truly) Local Broadcast Layer for Unreliable Radio Networks

Nancy Lynch
MIT CSAIL
lynch@csail.mit.edu

Calvin Newport
Georgetown University
cnewport@cs.georgetown.edu

Abstract

In this paper, we implement an efficient *local broadcast* service for the dual graph model, which describes communication in a radio network with both reliable and unreliable links. Our local broadcast service offers probabilistic latency guarantees for: (1) message delivery to all reliable neighbors (i.e., neighbors connected by reliable links), and (2) receiving *some* message when one or more reliable neighbors are broadcasting. This service significantly simplifies the design and analysis of algorithms for the otherwise challenging dual graph model. To this end, we also note that our solution can be interpreted as an implementation of the *abstract MAC layer* specification—therefore translating the growing corpus of algorithmic results studied on top of this layer to the dual graph model. At the core of our service is a *seed agreement* routine which enables nodes in the network to achieve "good enough" coordination to overcome the difficulties of unpredictable link behavior. Because this routine has potential application to other problems in this setting, we capture it with a formal specification—simplifying its reuse in other algorithms. Finally, we note that in a break from much work on distributed radio network algorithms, our problem definitions (including error bounds), implementation, and analysis do not depend on global network parameters such as the network size, a goal which required new analysis techniques. We argue that breaking the dependence of these algorithms on global parameters makes more sense and aligns better with the rise of ubiquitous computing, where devices will be increasingly working locally in an otherwise massive network. Our push for locality, in other words, is a contribution independent of the specific radio network model and problem studied here.

1 Introduction

In this paper, we implement an efficient *local broadcast* service in the dual graph radio network model [15, 17, 3, 8, 7, 11, 9], which captures wireless communication over both reliable and unreliable links. In more detail, the dual graph model describes a network with two graphs, one for each type of link. In each round, the network topology used for node communication is a combination of the edges from reliable link graph and some subset of the edges from the unreliable link graph, the latter chosen adversarially. As argued in our earlier studies of this setting, the inclusion of unpredictable links in formal models of radio communication is motivated by the ubiquity of such behavior in real networks (e.g., [23]).

Our local broadcast algorithm yields two types of probabilistic latency guarantees: a fast *progress bound*, which bounds the time for a node to receive *something* when one or more of its reliable neighbors (i.e., neighbors connected by reliable links) are transmitting, and a slower but still reasonable *acknowledgment bound*, which bounds the time for a sender to finish delivering a broadcast message to all of its reliable neighbors. The service we implement operates in an ongoing manner, which makes it suitable for use as an abstraction layer for developing higher-level distributed algorithms for unreliable radio networks. To this end, we note that our algorithm can be interpreted as an implementation of the *Abstract MAC Layer* specification [14, 16]. It follows that the growing corpus of results designed to run on top of this abstraction (e.g., [10, 20, 6, 13, 12, 5]) can be composed with our implementation, automatically porting these existing solutions to the dual graph model for the first time.

More generally speaking, we note that since 2009 [15] we have waged (algorithmic) battle with the complexities introduced by the presence of unpredictable link behavior in the dual graph model—describing upper and lower bounds for a variety of problems [15, 17, 3, 8, 7, 11, 9]. This paper can be seen as the culmination of this half-decade of effort, in which we integrate what we have learned into a powerful abstraction that renders this realistic but difficult setting tractable to a wider community of algorithm designers.

True Locality. Before proceeding to the details or our results, we must first emphasize an important property of our service: its implementation is "truly local," by which we mean that its specification, time complexity, and error bounds are expressed independent of global parameters such as network size. To make this work: we define our correctness and performance properties locally, in terms of individual nodes, rather than globally, in terms of all the nodes; we express our time complexity bounds with respect to local properties, such as bounds on local neighborhood size, not the full network size, n; and we capture our error probabilities as generic ϵ parameters, rather than the common approach of bounding the error in terms of $(1/n^c)$, for some constant $c \geq 1$. Though locality of this type has been well-studied in other network models (e.g., [19, 18]), it is less understood in the wireless setting—a deficiency we believe must be addressed. There are two justifications for this belief.

First, the common practice of seeking "high probability" (i.e., an error probability bound of the form n^{-c}) seems unnatural for most deployment scenarios—why should you have to grow your network size to decrease your error probability?[1] Of course, by instead introducing a generic error parameter, ϵ, as we do in this paper, we are *not* eliminating the possibility of high probability error bounds when useful, as you can simply set $\epsilon = (1/n^c)$ if needed. But we believe that one should try to postpone considering such dependence until it is really necessary. Second, as we see an increasing emphasis on an *Internet of Things* style ubiquity in the wireless setting [1], global properties such as network size can grow to a massive scale. In studying local algorithms for such scenarios, it is important that we separate time complexity and error guarantees from global parameters and instead reorient them toward the relevant local conditions. This paper provides an example of what is required for a reorientation of this type.

Results. Our local broadcast service is parametrized by two time bounds, t_{ack} and t_{prog}, and an error bound, ϵ. It assumes a dual graph model with an oblivious link scheduler (i.e., decisions about which unreliable links to include in the topology in each round are made at the beginning of the execution), and a natural geographic constraint that requires all nodes within distance 1 be connected by a reliable edge, and no nodes at distance more than $r \geq 1$ be connected by an unreliable edge. (We typically assume r is constant and therefore omit it in our result summary below. For the sake of completeness, however, we keep r as a parameter in our analysis during all steps leading to these results.) Our service guarantees: (1) for each broadcast of a message m by a node u, with probability at least $1 - \epsilon$, every reliable neighbor of u will receive m within t_{ack} rounds; and (2) for a given receiver v and span of t_{prog} rounds, such that at least one reliable neighbor of v is broadcasting throughout the entire span, the probability that v *fails* to receive at least one message during these rounds is no more than ϵ. We present an algorithm that takes ϵ as a parameter and implements this service for values of t_{ack} and t_{prog} bounded as follows:

- $t_{prog} = O\left(\log \Delta \log \left(\frac{\log^4 \Delta}{\epsilon}\right)\right)$
- $t_{ack} = O\big(\Delta \log(\Delta/\epsilon) \log \Delta \log\left(\frac{\log^4 \Delta}{\epsilon}\right)\left(\frac{1}{1-\epsilon}\right)\big)$

where Δ is an upper bound on node degree. We emphasize that that these results are near optimal, as even

[1]The most likely answer for why these high probability bounds persist is that they make life easier for the algorithm designer. In particular, if a property holds with high probability in n, a basic union bound provides that the property holds for all nodes in a network for all rounds of any reasonable length execution, which greatly simplifies subsequent analysis. We believe, however, that the algorithm designer should do more work to make the treatment of error probability more natural to the practitioner using the algorithms.

in the absence of unreliable links: (1) any progress bound (which reduces to symmetry breaking among an unknown set of nodes) requires logarithmic rounds (e.g., [21]); and (2) any acknowledgement bound requires at least Δ rounds in the worst case (imagine a receiver u neighboring Δ broadcasters: u can only receive one message per round, delaying one of these broadcasters by at least Δ rounds).

Discussion. A core difficulty in solving local broadcast in the dual graph model is the presence of the unreliable links, which are included or excluded in the network topology according to an arbitrary link schedule (see the model section below for details). To understand this difficulty, recall that the standard strategy for broadcast in a radio network is to cycle through a fixed schedule of geometrically decreasing broadcast probabilities [2]. The intuition for this fixed schedule approach is that for each receiver, one of these probabilities will be well-suited for the local contention among its nearby broadcasters. In the dual graph model, however, there is no fixed amount of local contention: the link schedule can effectively change this amount at each receiver at each round by changing the edges included in the network topology. It is possible, for example, that the link schedule was constructed with the intent of thwarting this fixed schedule strategy by including many links (i.e., increasing contention) when the schedule selects high probabilities, and excluding many links (i.e., decreasing contention) when the schedule selects low probabilities.

To overcome this difficulty, we use as a starting point the general strategy we originally identified in [11]: permute the broadcast probability schedule *after* the execution begins (and therefore, *after* the link schedule has already been generated) to regain independence from the topology. The challenge in permuting a broadcast probability schedule, however, is coordinating the nodes sufficiently that they can apply the same permutation. This creates a chicken and egg problem: to share information among processes requires that we solve broadcast, but we are sharing this information to help solve broadcast. Our solution is to instead solve a form of loose coordination we call *seed agreement.* This problem assumes each participant generates a *seed* (for our purposes, this seed will be random bits for use in generating a probability permutation), and then attempts to convince nearby nodes to *commit* to its seed. A solution to this problem must guarantee that every node commits to some nearby seed (perhaps its own), and, crucially, that there are not too many unique seeds committed in any node's neighborhood (in this paper, we achieve a bound on the order of $\log(1/\epsilon)$, for error probability ϵ). If nodes subsequently use these seeds to permute their broadcast probability schedules (thereby gaining independence from the link schedule), we are assured that there are not *too many* different schedules in a given neighborhood. An extra conflict resolution mechanism is then introduced to our broadcast strategy to help resolve contention among these competing schedules.

We note that the seed agreement subroutine provides a general strategy for taming adversarial link scheduling, and is therefore of independent interest. Accordingly, in this paper we provide a standalone formal specification for the problem that is satisfied by our algorithm. This simplifies the process of subsequently integrating our seed agreement solution into other algorithms.

Finally, we note that to solve these problems in the absence of global parameters such as n requires the introduction of new and non-trivial proof techniques, also of independent interest for the general pursuit of true locality in radio network algorithms. For example, in analyzing our seed agreement subroutine, we could not simply assume that certain key parameters involving local contention hold for the whole network, as this would require a dependence on n. We instead established a *region of goodness* surrounding the target node in our analysis. Using a careful induction on algorithm steps, we showed that although the guaranteed radius of this region must contract as time advances (due to the influence of nodes outside the region for whom we make no assumptions), this contraction is slow enough that our target node safely completes its computation in time. Similarly, in analyzing the local broadcast routines that leverage the seed agreement bits, we had to leverage a new notion of "high probability" that is defined with respect to node degree, not network size (this is the source of the $\log(\Delta/\epsilon)$ factors in the t_{ack} bound described above).

Related Work. The dual graph model of unreliable radio communication was introduced by Clementi et al. [4], and subsequently given the name "dual graph" by Kuhn et al. [15]. It has since been well-studied [15, 17, 3, 8, 7, 11, 22]. Under the pessimistic assumption of an adaptive adversary (as oppose to the oblivious adversary considered in this paper), we previously explored bounds for global broadcast [15, 17], local broadcast [8], and graph structuring algorithms [3, 22]. In [11], we studied the impact of different link scheduler models by proving that some of the more pessimistic bounds from our previous work depended on the assumption that the link schedule was constructed by an adaptive adversary. Of particular relevance to this paper, we proved in [11] that local broadcast with efficient progress is impossible with an adaptive link scheduler of this type, but is feasible with an oblivious link schedule. To establish this latter point, we designed and analyzed a one-shot local broadcast algorithm that offers a progress guarantee (i.e., every node that neighbors a broadcaster will get *some* message quickly) but no reliability guarantees (i.e., no particular message is guaranteed to be delivered). The algorithm in [11] introduced the basic ideas that we developed into the seed agreement specification and $SeedAlg$ algorithm presented in this paper. We also note that all results [11] depend on global parameters, whereas here we invest significant effort in gaining true locality.

The abstract MAC layer [14, 16] is an approach to designing wireless network algorithms that defines an abstraction that captures the main guarantees of most wireless link layers, and then divides the task of algorithm design into two efforts: (1) describing and analyzing algorithms that run on top of the abstraction, and (2) implementing the abstraction in specific low-level wireless models. Our local broadcast problem was defined with the standard parameters of a (probabilistic) abstract MAC layer in mind. Our algorithm, therefore, can be interpreted as a strategy for implementing this layer in the dual graph radio network model, and therefore providing a way to translate to the dual graph low level model the growing corpus of algorithms designed and analyzed onto of the abstract MAC layer [10, 20, 6, 13, 12, 5]. We note, however, that the translation from our algorithm to an abstract MAC layer implementation is not immediate, as some (presumably straightforward) work will be required to mediate between our definition, expressed in terms of low-level details like rounds and receiving messages, and the higher level specification of the abstract MAC layer, which is usually expressed only in terms of the ordering and timing of input and output events.

2 The Dual Graph Model

We use a radio network model based on dual graphs, which describes randomized algorithms executing in a synchronous multihop radio network with both reliable and unreliable links. The model describes the network topology with a dual graph (G, G'), where $G = (V, E)$, $G' = (V, E')$, and $E \subseteq E'$, where E describes reliable links and $E' \setminus E$ describes unreliable links. We use n to denote $|V|$, the number of vertices in the graphs. For $u \in V$, we write $N_G(u)$ ($N_{G'}(u)$) to denote u's immediate neighbors in G (G'), not including u itself. We assume two degree bounds: Δ, an upper bound on $|N_G(u) \cup \{u\}|$, and Δ', an upper bound on $|N_{G'}(u) \cup \{u\}|$, defined over every u.

An *embedding* of a (finite) set V of graph vertices in the Euclidean plane is simply a mapping $emb : V \rightarrow \mathbb{R}^2$; this provides a pair of (x, y) coordinates for each vertex V. If emb is an embedding of the vertices V of a dual graph (G, G') and r is a real number, $r \geq 1$, then we say that (G, G') is *r-geographic* with respect to emb provided that, for every $u, v \in V, u \neq v$, the following conditions hold (where d represents Euclidean distance):

1. If $d(emb(u), emb(v))) \leq 1$ then $\{u, v\} \in E$.
2. If $d(emb(u), emb(v)) > r$, then $\{u, v\} \notin E'$.

In other words, nearby vertices must be neighbors in G, and distant vertices cannot even be neighbors in G', but vertices in the grey zone represented by the intermediate distances in $(1, r]$ might or might not be neighbors in G or G'. We say that (G, G') is *r-geographic* provided that there exists an embedding emb of the vertex set V such that (G, G') is r-geographic with respect to emb. We sometimes also say that (G, G') is *geographic* provided that there exists a real $r \geq 1$ such that (G, G') is r-geographic.

We assume that the dual graphs we consider are r-geographic, for some particular r, which we fix for the rest of the paper. Moving forward, fix an space I. An *algorithm* is an injective mapping $proc()$ from I to some set of *processes*, which are some type of probabilistic timed automata that model wireless devices. Thus, $proc(i)$ denotes the process with id i. We assume that each process knows (e.g., has in a special component of its initial state) its own id, and also knows the quantities Δ, and Δ'. Notice, we do not assume processes know n (as is typical in such models) as we seek problem definitions and solutions that operate independently of the network size. A *process assignment* for a dual graph (G, G') and id space I is an injective mapping $id()$ from V to I, that assigns a different id to each graph vertex. The two mappings, $proc$ and id, together serve to assign a distinct process to each graph vertex. That is, $proc(id(u))$ is the process assigned to graph vertex u. To simplify terminology, we often write *node* u to indicate $proc(id(u))$ or *process* i to indicate $proc(i)$. We assume that processes do not know the $id()$ mapping in advance.

An execution of an algorithm in a given dual graph network topology (G, G') proceeds in synchronous rounds $1, 2, \ldots$. In each round t, each node decides whether to transmit a message or receive, based on its process definition; this might involve a random choice. The communication topology in round t consists of the edges in E plus an arbitrary subset of the edges in $E' \setminus E$. This subset, which can change from round to round, is determined by an adversary that we call a *link scheduler* (see below). Once the topology is fixed for a given round, we use the following standard collision rules to determine communication: node u receives a message m from node v in round t, if and only if: (1) u is receiving; (2) v is transmitting m; and (3) v is the only node transmitting among the neighbors of u in the communication topology chosen by the link scheduler for round t. If node u does not receive a message, then we assume that it receives a special "null" indicator $\perp$: that is, we assume no collision detection.

We now formalize the notion of a *link scheduler*: the entity responsible for resolving the non-determinism concerning which edges from $E' \setminus E$ are added to the topology in each round. Formally, we define a link scheduler to be a sequence $\mathbb{G} = G_1, G_2, G_3, \ldots$, where each G_t (also denoted $\mathbb{G}[t]$) is the graph used for the communication topology in round t. We assume each G_t is allowable given our above model definition.[2] We assume the link scheduler for a given execution is specified at the beginning of the execution. Notice, this definition implies oblivious behavior concerning the network dynamism, as all decisions on the topology are made at the beginning of an execution.

The other relevant source of non-determinism in our model is the *environment* which we use to provide inputs and receive outputs as required by a specific problem (when relevant). For example, in solving local broadcast, an environment provides the messages to broadcast, whereas for a problem like consensus, it provides the initial values. The details of what defines a *well-formed* environment is specified on a problem-by-problem basis. As with the scheduler, when analyzing an execution we first fix the environment for the execution. Though it is possible to conceive of an environment as a probabilistic entity, for the sake of simplicity, the environments we consider in this paper are all deterministic (i.e., once you fix an environment for an execution, all non-determinism regarding inputs is resolved). To formally model the interaction with an environment, we break down the synchronous steps each process takes within a round as follows: first all processes receive inputs (if any) from the environment, next all processes that decide to transmit do so, they then all receive, and finally, they generate outputs (if any) which are processed by the environment to end the round.

We call the combination of a dual graph, process assignment, link scheduler, and environment a *configuration.* Notice, a configuration resolves all relevant model and problem nondeterminism. It follows that a configuration combined with a probabilistic algorithm defines a distribution over possible executions (which we sometimes call the execution tree). When we say an algorithm satisfies a certain property with a given probability, we mean that for all allowable configurations, in the execution tree that results from combining

[2]That is, it is a graph that includes all the nodes and edges of G with (perhaps) some edges from $E' \setminus E$ also included.

the algorithm with the configuration, the property is satisfied with that probability.

3 Seed Agreement

The *seed agreement* problem provides a loose form of coordination: each participating node u generates a *seed* s from some known seed domain S, and then eventually commits to a seed generated by a node in its neighborhood (perhaps its own). The safety goal is to bound the number of unique seeds committed in any given neighborhood by a sufficiently small factor δ, while the liveness goal is to do so in a minimum of rounds. In this section, we provide a dual graph algorithm that yields a bound δ that is roughly $O(\log(\frac{1}{\epsilon}))$, and that operates within time that is polynomial in $\log(\Delta)$ and $\log(\frac{1}{\epsilon})$, with (provided) error probability ϵ.

In Section 4, we use seed agreement as a crucial subroutine in our local broadcast service implementation. It is potentially useful, however, to any number of problems in the dual graph model (and elsewhere), so we take our time here to first provide a careful formal specification, which we then satisfy with a new dual graph algorithm. The analysis of our algorithm was rendered particularly tricky by our goal of avoiding dependence on global parameters such as n, and provides some of the main technical contributions of this paper. Due the long length of this analysis, we provide below only the problem definition, our algorithm description, and our main theorem—deferring the analysis details to Appendix B. We point the interested reader to Appendix B.4 as a good example of how we are forced to compensate for our inability to establish global properties with a union bound (which would require a dependence on n). The strategy deployed in this section is to bound the rate at which a region of "goodness" (i.e., sufficiently bounded contention) surrounding our target node contracts as the node races toward termination.

3.1 The Seed Agreement Problem

Fix a finite *seed domain* S. We specify the problem as $Seed(\delta, \epsilon)$, where δ is a positive integer representing the seed partition bound, and ϵ is a small nonnegative real representing an error probability. This specification describes correctness for a system based on some arbitrary system configuration, running according to our execution definition. The specification has no inputs. Its outputs are of the form $decide(j, s)_u$, where $j \in I$, $s \in S$, and $u \in V$. This represents a decision by the node at graph vertex i to commit to the seed s proposed by the node with id j (in the following, we call j the *owner* of seed s). We begin with two basic non-probabilistic conditions on the outputs; these must hold in every execution:

1. *Well-formedness:* In every execution, for each vertex u, exactly one $decide(*, *)_u$ occurs.

2. *Consistency:* In every execution, for each pair of vertices u_1, u_2, if $decide(j, s_1)_{u_1}$ and $decide(j, s_2)_{u_2}$ both occur, then $s_1 = s_2$.

That is, if outputs contain the same owners then they also contain the same seeds. The two remaining conditions are probabilistic. To talk about probabilities of events, we must first specify the probability distribution. As noted in Section 2, the combination of the system configuration fixed above and a given seed agreement algorithm defines a distribution on executions. We state our remaining properties in terms of this distribution. In more detail, we start by considering an agreement property. Let $B_{u,\delta}$ be the event (in the probability space of executions) that at most δ distinct ids appear as seed-owners in $decide$ outputs at nodes in $N_{G'}(u) \cup \{u\}$.

3. *Agreement:* For each vertex u, $Pr(B_{u,\delta}) \geq 1 - \epsilon$.

Note that we state Condition 3 for each vertex u separately, rather than in terms of all vertices, as in [11]. This change is needed for expressing costs in terms of local parameters.

We now express independence of the choices of seed values corresponding to different owners. An *owner mapping* M_o is a mapping from V to I, that is, an assignment of an (owner) id to each vertex. A *seed mapping* M_s is a mapping from V to S, that is, an assignment of a seed to each vertex. We say that a seed mapping M_s is *consistent* with an owner mapping M_o provided that, if two vertices have the same owner, then also have the same seed. That is, if $M_o(u) = M_o(v)$ then $M_s(u) = M_s(v)$. Let Own_{M_o} be the event in the probabilistic execution that M_o is the owner mapping.

4. *Independence:* Suppose that M_o is an owner mapping and M_s is a seed mapping, where M_s is consistent with M_o. Suppose that $\Pr(Own_{M_o}) > 0$. Then, conditioned on Own_{M_o}, the probability that M_s is the seed mapping that appears in the execution is exactly $(\frac{1}{|S|})^{|range(M_o)|}$.

Condition 4 says that the probability of each consistent seed mapping is just what it would be if the seed mapping were determined in the following way: All processes first choose a seed from S, uniformly at random. Then after every process chooses a seed owner, it also adopts the associated seed value.

3.2 A Seed Agreement Algorithm

We now describe our seed agreement algorithm, $SeedAlg$, which takes as its sole parameter, an error bound, ϵ_1. We will show, in Theorem 3.1, that this algorithm implements $Seed(\delta, \epsilon)$ for values of δ and ϵ that depend on ϵ_1. Its main strategy is to hold aggressive local leader elections that yield bounded safety violations (i.e., multiple nearby leaders). In the following description, we assume for simplicity that Δ is a power of 2. We also use a "sufficiently large" constant c_4 for the phase length.

Algorithm $SeedAlg(\epsilon_1)$, for process i at graph vertex u, where $0 < \epsilon_1 \leq \frac{1}{4}$:

The algorithm uses $\log \Delta$ *phases*, each consisting of $c_4 \log^2(\frac{1}{\epsilon_1})$ rounds.
Process i maintains a local variable containing its "initial seed" in S, which it chooses uniformly at random from the seed domain S. It also keeps track of its *status* $\in$ {"$active$", "$leader$", "$inactive$"}, and the current phase number and round number.

Now we describe process i's behavior in any particular phase $h \in \{1, \dots, \log \Delta\}$. If $status = active$ at the beginning of phase h, then process i becomes a leader; i.e., sets $status$ to $leader$, with probability $2^{-(\log \Delta - h + 1)}$. Thus, it uses probabilities: $\frac{1}{\Delta}, \frac{2}{\Delta}, \dots, \frac{1}{4}, \frac{1}{2}$, as it progresses through the phases.

If process i becomes a leader at the start of phase h, it immediately outputs $decide(i, s)$, where s is its initial seed. Then, during the remaining rounds of the phase, process i broadcasts (i, s) with probability $\frac{1}{\log(\frac{1}{\epsilon_1})}$ in each round. At the end of the phase, it becomes inactive.

If process i is active but does not become a leader at the start of phase h, then it just listens for the entire phase. If it receives a message containing a pair (j, s), then it immediately outputs $decide(j, s)$ and becomes inactive. If it receives no messages during this phase, then it remains active.

If process i completes all phases and is still active, then it outputs $decide(i, s)$, where s is its initial seed.

3.3 Correctness of $SeedAlg$

The analysis of $SeedAlg$ contained in Appendix B culminates with the main theorem below. For the following, recall that r is the value used in defining the r-geographic property assumed of our dual graph (Section 2), and Δ is the maximum node degree in G.

Theorem 3.1. *$SeedAlg(\epsilon_1)$ satisfies the $Seed(\delta, \epsilon)$ specification, where δ is $O(r^2 \log(\frac{1}{\epsilon_1}))$, and $\epsilon = O(r^4 \log^4(\Delta)(\epsilon_1)^{c^{r^2}})$, where c is some constant, $0 < c < 1$. The algorithm takes $O((\log \Delta) \log^2(\frac{1}{\epsilon_1}))$ rounds.*

4 Local Broadcast

We now define our local broadcast service, then describe and analyze an efficient solution which uses the $SeedAlg$ algorithm from Section 3 as a key subroutine. As before, due to the length of our analysis, we describe the algorithm, problem, and main correctness theorem below, but defer details to Appendix C.

4.1 The Local Broadcast Problem

The local broadcast problem described here requires nodes to implement an ongoing probabilistic local communication service with timing and reliability guarantees. In more detail, we call the problem $LB(t_{ack}, t_{prog}, \epsilon)$, where $t_{ack} \geq t_{prog} \geq 1$ are integer round bounds, and ϵ is a small real representing an upper bound on the error probability. To define the problem, we must first fix the underlying dual graph network in which it is being solved: $(G = (V, E), G' = (V, E'))$. We then define a set $\mathcal{M}_u$ of possible messages for each $u \in V$. For simplicity, we assume these sets are pairwise disjoint. Let $\mathcal{M} = \bigcup_u \mathcal{M}_u$ be the set of all possible messages. Every node $u \in V$ has a $bcast(m)_u$ input and $ack(m)_u$ output, for each $m \in \mathcal{M}_u$. Node u also has a $recv(m')_u$ output for each $m' \in \mathcal{M}$.

We now restrict the behavior of the environments we consider for this problem. In more detail, we assume that (1) the environment generates each input at a given node u at most once per execution (i.e., each message it passes a node to broadcast is unique), and (2) if the environment generates a $bcast(m)_u$ input at u at some round r, it must then wait until u subsequently generates a $ack(m)_u$ output (if ever) before it can generate another $bcast$ input at u. To simplify analysis, we restrict our attention to deterministic environments. Therefore, we can assume the environment is modeled as synchronous deterministic automaton that receives the nodes' ack outputs as input, and generates their $bcast$ inputs as its output.

In the following, we say a node u is *actively broadcasting* m *in round* r, if node u received a $bcast(m)_u$ input in some round $r' \leq r$, and through round r, node u has not yet generated a subsequent $ack(m)_u$ output. Similarly, we say u is *active* in a given round if there is some message that u is actively broadcasting during this round. The problem places deterministic and probabilistic constraints on the nodes' output behavior. We begin with the deterministic constraints.

In every execution, the following must always hold:

1. *Timely Acknowledgement.* If node u receives a $bcast(m)_u$ input in round r, it will generate a single corresponding $ack(m)_u$ output in the interval r to $r + t_{ack}$. These are the only ack outputs that u generates.

2. *Validity.* If node u performs a $recv(m)_u$ output in some round r, then there exists some $v \in N_{G'}(u)$ such that v is actively broadcasting m in round r.

Recall that if we fix some configuration and an algorithm, the combination yields a well-defined probability distribution (equiv., execution tree) over executions of the algorithm in this configuration. To aid our probabilistic constraint definitions, we first introduce some notation for discussing an execution tree determined by a configuration. When considering any execution from such a tree, we can partition time into *phases* of length t_{prog} starting in the first round. We number these $1, 2, \ldots$. We use the terminology *phase* *i prefix* to describe a finite execution prefix that includes every round up to the beginning of phase i (i.e., it does not include the first round of phase i but does include the last round—if any—before phase i begins). For a given execution tree, phase i prefix α in this tree, and node u, let A^u_α describe the set of t_{prog}-round extensions of α in which there is a G-neighbor of u that is active throughout every round of phase i, and let B^u_α describe the set of t_{prog}-round extensions where u receives at least one message $m_v \in M_v$ from a node v in a round r such that v is actively broadcasting m_v in r.

We now define two probabilistic constraints that must hold for every configuration:

1. *Reliability.* For every configuration, node u, and r-round execution prefix such that u receives a $bcast(m)_u$ input at the beginning of round r: the probability that every $v \in N_G(u)$ generates a $recv(m)_u$ output before u's corresponding $ack(m)_u$ output, is at least $1 - \epsilon$.

   (Notice: this property leverages the *timely acknowledgment* property which tells us that in every extension of this prefix, u generates an $ack(m)_u$ output wihin t_{ack} rounds.)

2. *Progress.* For every configuration, node u, phase i, and phase i prefix α in the resulting execution tree: $\Pr(B^u_\alpha \mid A^u_\alpha) \geq 1 - \epsilon$.

4.2 A Local Broadcast Algorithm

We now describe $LBAlg$: our solution to the local broadcast problem. This description makes use of several constants that we detail in Appendix C.1. For our purposes here, it is sufficient to know the following regarding their definition and size: ϵ_1 is the desired error bound for LB, κ describes the number of bits needed to resolve one phases's worth of common random choices, T_s is an integer in $O\big(\log \Delta \log^2(\frac{1}{\epsilon_2})\big)$, T_{prog} is an integer in $O\big(r^2 \log(\frac{1}{\epsilon_1}) \log(\frac{1}{\epsilon_2}) \log \Delta\big)$, T_{ack} is an integer in $O\big(\frac{\Delta \log(\Delta/\epsilon_1)}{(1-\epsilon_1)}\big)$, and ϵ_2 is an error probability defined to be sufficiently small that $SeedAlg$ solves seed agreement with an error bound $\leq \epsilon_1/2$.

Algorithm $LBAlg(\epsilon_1)$, for process i at vertex u for some real $\epsilon_1, 0 < \epsilon_1 \leq \frac{1}{2}$.

Node u partitions rounds into *phases* of length $T_s + T_{prog}$ rounds. We label these phases $1, 2, 3, \ldots$. During each phase, u can be in one of two states: *receiving* or *sending*. Node u begins the execution in the receiving state. After receiving a $bcast(m)_u$ input, u will spend the next T_{ack} full phases in the sending state (if it receives the $bcast$ input in the middle of a phase, it waits until the beginning of the next phase to switch to the sending state). At the end of the last round of the last of these T_{ack} phases, node u generates an $ack(m)_u$ output, and then returns to the receiving state.

We now define what happens during these phases. At the beginning of each phase, regardless of u's state, it executes $SeedAlg(\epsilon_2)$ as a subroutine, using the seed set $S_\kappa = \{0,1\}^\kappa$; i.e., the set describing every bit sequence of length κ. Let $s^{(j)}_u$ be the seed that node u commits in the beginning of phase j. We call the rounds spent at the beginning of a phase running $SeedAlg$ the *preamble* of the phase, and the remaining rounds the *body* of the phase.

Node u's behavior during the body of a given phase j depends on its state. If it is in the receiving state, it simply receives during each of these rounds. If during one of these rounds, node u receives a message m' that it has not yet previously received, it generates a $recv(m')_u$ output.

On the other hand, if u is in the sending state for this phase, during each of the body rounds, it does the following:

1. Node u consumes $\lceil \log{(r^2 \log{(\frac{1}{\epsilon_2})})} \rceil$ new bits from its seed $s_u^{(j)}$. If all of these bits are 0 (which occurs with probability $a \cdot \frac{1}{r^2 \log{(\frac{1}{\epsilon_2})}}$, for some $a \in [1, 2)$) it sets its status to *participant*, otherwise it sets its status to *non-participant*.
2. If u is a non-participant, it receives.
3. If u is a participant, it next consumes $\log \log \Delta$ new bits from $s_u^{(j)}$. Let b be the value in $[\log \Delta]$ specified by these bits. The node then uses an independent (with respect to the other processes) local source of randomness (i.e., *not* bits from $s_u^{(j)}$), to generate b bits with uniform randomness. If all b bits are 0 (which occurs with probability 2^{-b}), u broadcasts its message.

As with the receiving state, if during any of these body rounds, node u receives a message m' that it has not yet previously received, it generates a $recv(m')_u$ output.

The above algorithm divides rounds into phases and then runs a seed agreement algorithm at the beginning of each phase to synchronize shared random bits for the remainder of the phase. Notice that there is nothing fundamental about this frequency of seed agreements. In some settings, it might make sense to run the agreement protocol less frequently, and generate seeds of sufficient length to satisfy the demands of multiple phases. Such modifications do not change our worst-case time bounds but might improve an average case cost or practical performance.

4.3 Correctness of $LBAlg$

In Appendix C, we analyze $LBAlg$, culminating in the following theorem:

Theorem 4.1. *$LBAlg(\epsilon_1)$ solves the $LB(t_{ack}, t_{prog}, \epsilon_1)$ problem for:*

- $t_{prog} = T_s + T_{prog} = O\left(r^2 \log \Delta \log{(\frac{r^4 \log^4 \Delta}{\epsilon_1})}\right)$
- $t_{ack} = (T_{ack} + 1)(T_s + T_{prog}) = O\big(r^2 \Delta \log{(\Delta/\epsilon_1)} \log \Delta \log{(\frac{r^4 \log^4 \Delta}{\epsilon_1})}(\frac{1}{1-\epsilon_1})\big)$

Below is the core lemma on which we build our proof of Theorem 4.1. This lemma bounds the behavior of $LBAlg$ within the scope of a single phase. To do so, we first introduce some useful notation. For a given phase i of an execution, let B_i be the set of nodes that are in sending status during phase i, and $R_i = N_G(B_i)$ be the set of nodes that neighbor B_i in G. Notice, because sending status is fixed for the duration of a given phase, both B_i and R_i are determined at the beginning of phase i and cannot change during the phase. Also recall from the model definitions that Δ' bounds the maximum degree in G'. Using this notation, we specify and prove the following key probabilistic behavior:

Lemma 4.2. *Fix some phase i and an execution prefix through the $(j-1)^{th}$ body round of this phase, for some $j \in \{2, ..., T_{prog}\}$. Fix nodes u and v, where $u \in R_i$ and $v \in N_G(u) \cap B_i$. Assume the call to $SeedAlg$ at the beginning of phase i in this prefix satisfies $B_{u,\delta}$. Let p_u be the probability that u receives some message in the j^{th} round, and let $p_{u,v}$ be the probability that u receives a message* from *v in this round. It follows that:*

- $p_u \geq \frac{c_2}{r^2 \log{(\frac{1}{\epsilon_2})} \log \Delta}$
- $p_{u,v} \geq p_u/\Delta'$

The full proof for this lemma (see Appendix C.2) works through the details of the following intuition. Under the assumption that the call to $SeedAlg$ at the beginning of this phase is successful, the neighbors of u can be partitioned into at most $\delta = O(r^2 \log(\frac{1}{\epsilon_2}))$ groups, such that each group shares the same seed (see Theorem 3.1). Because u has a reliable neighbor that is transmitting (by assumption), at least one such group S_u contains one or more G neighbors. By the definition of $LBalg$, with probability $\Theta((r^2 \log(\frac{1}{\epsilon_2}))^{-1})$, this group will be the only group to decide to be a participant group in this round (these decisions are uniform within a group since they are based on a common seed). Assuming this happens, we next fix the edges added to the topology for this round by the link schedule. Let ℓ be the number of G edges from S_u to u in this topology. We note that there is a "successful" probability for this value of ℓ (i.e., anything close to $1/\ell$), and that members from S_u, using bits from their common seed, will select this successful value with probability $1/\log\Delta$. At this point, with an additional constant probability, exactly one of these ℓ neighbors of u will broadcast. Combining these various probabilities provides the p_u bound from above. The $p_{u.v}$ bound follows from the uniformity and independence with which the ($\leq \Delta'$) nodes in S_u make their broadcast decisions.

5 Conclusion

In this paper, we described and analyzed an ongoing local broadcast service for the dual graph model. This service hides the complexities introduced by unpredictable link behaviors and therefore has the potential to significantly simplify the development of distributed algorithms for this challenging setting. As noted in the introduction, we see this result as the culmination of a half-decade struggle [15, 17, 3, 8, 7, 11, 9] to tame the realistic but difficult link unpredictability at the core of the dual graph model definition.

Our solution can also be adapted to implement the abstract MAC layer specification [14, 16], allowing existing results for this abstraction to translate to the dual graph model (e.g., [10, 20, 6, 13, 12, 5]). Though we leave the details of this adaptation to future work, we note that it would likely be straightforward, with the main effort focused on aligning our local broadcast problem definition—which depends on low level model details, such as rounds and receiving messages—with the higher level of the abstract MAC layer, which is specified in terms of the timing and ordering of input and output events.

Finally, we emphasize that our commitment to *truly local* algorithms, which required us to avoid global parameters in problem definitions, algorithm strategies, and analysis, is of standalone interest to those studying radio network algorithms. As we argued at this paper's opening, a local perspective provides more flexibility to practitioners, and will become increasingly necessary as network sizes grows.

6 Acknowledgments

This research is supported in part by: Ford Motor Company University Research Program, NSF Award CCF-1320279, NSF Award CCF-0937274, NSF Award CCF-1217506, NSF CCF-0939370, and AFOSR Award Number FA9550-13-1-0042.

References

[1] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A Survey. *Computer Networks*, 54(15):2787–2805, 2010.

[2] R. Bar-Yehuda, O. Goldreigch, and A. Itai. On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap between Determinism and Randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[3] K. Censor-Hillel, S. Gilbert, F. Kuhn, N. Lynch, and C. Newport. Structuring unreliable radio networks. *Distributed Computing*, 27(1):1–19, 2014.

[4] 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.

[5] A. Cornejo, N. Lynch, S. Viqar, and J. L. Welch. Neighbor Discovery in Mobile Ad Hoc Networks Using an Abstract MAC Layer. In *Annual Allerton Conference on Communication, Control, and Computing*, 2009.

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

[7] M. Ghaffari. Bounds on contention management in radio networks. Master's thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, February 2013.

[8] M. Ghaffari, B. Haeupler, N. Lynch, and C. Newport. Bounds on contention management in radio networks. In M. K. Aguilera, editor, *Distributed Computing: 26th International Symposium (DISC 2012), Salvador, Brazil, October, 2012*, volume 7611 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2012.

[9] M. Ghaffari, E. Kantor, N. Lynch, and C. Newport. Multi-message broadcast with Abstract MAC layers and unreliable links. In *Proceedings of the 33nd Annual ACM Symposium on Principles of Distributed Computing (PODC'14)*, pages 56–65, Paris, France, July 2014.

[10] M. Ghaffari, E. Kantor, N. Lynch, and C. Newport. Multi-message broadcast with abstract mac layers and unreliable links. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2014.

[11] M. Ghaffari, N. Lynch, and C. Newport. The cost of radio network broadcast for different models of unreliable links. In *Proceedings of the 32nd Annual ACM Symposium on Principles of Distributed Computing*, pages 345–354, Montreal, Canada, July 2013.

[12] M. Khabbazian, F. Kuhn, D. Kowalski, and N. Lynch. Decomposing Broadcast Algorithms Using Abstract MAC Layers. In *Proceedings of the International Workshop on Foundations of Mobile Computing*, 2010.

[13] M. Khabbazian, F. Kuhn, N. Lynch, M. Medard, and A. ParandehGheibi. MAC Design for Analog Network Coding. In *Proceedings of the International Workshop on Foundations of Mobile Computing*, 2011.

[14] F. Kuhn, N. Lynch, and C. Newport. The Abstract MAC layer. In *Distributed Computing, 23rd International Symposium on Distributed Computing (DISC 2009), Elche, Spain, September 2009*, volume 5805 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2009.

[15] F. Kuhn, N. Lynch, and C. Newport. Brief announcement: Hardness of broadcasting in wireless networks with unreliable communication. In *Proceedings of the 28th Annual ACM Symposium on the Principles of Distributed Computing (PODC 2009)*, Calgary, Alberta, Canada, August 2009.

[16] F. Kuhn, N. Lynch, and C. Newport. The Abstract MAC layer. *Distributed Computing*, 24(3-4):187–206, November 2011. Special Issue for *DISC 2009: 23rd International Symposium on Distributed Computing*.

[17] F. Kuhn, N. Lynch, C. Newport, R. Oshman, and A. Richa. Broadcasting in unreliable radio networks. In *Proceedings of the 29th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 336–345, Zurich, Switzerland, July 2010.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2004.

[19] M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[20] C. Newport. Consensus with an abstract mac layer. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2014.

[21] C. Newport. Lower bounds for radio networks made easy. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2014.

[22] C. Newport. Lower bounds for structuring unreliable radio networks. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2014.

[23] C. Newport, D. Kotz, Y. Yuan, R. S. Gray, J. Liu, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *Simulation*, 83(9):643–661, 2007.

Appendix

A Mathematical Preliminaries

Here we define and isolate some important properties and concepts used in the analyses that follow. We also identify some mathematical facts that prove useful in these efforts.

A.1 Region Partitions

The arguments in [11] is based on a partition of the graph vertices. Instead, our work here is mainly based on a partition of the Euclidean plane into convex regions. We consider partitions of the plane that satisfy certain constraints on the size of a region and on the number of regions in a limited-size area of the plane.

Let $\mathcal{R}$ be any partition of the plane into (nonempty) convex regions and let $r \geq 1$ be a real number. We define the *region graph* for $\mathcal{R}$ and r, $G_{\mathcal{R},r}$, as follows. The vertices are just the regions in $\mathcal{R}$. An edge is included between regions R and R', $R \neq R'$, exactly if $d(p,q) \leq r$ for some $p \in R$ and $q \in R'$. We say that $(\mathcal{R}, r)$ is *f-bounded*, where f is a monotone nondecreasing function from nonnegative integers to nonnegative reals, provided that both of the following hold.

1. For every region $R \in \mathcal{R}$, and for every pair of points $p, q \in R$, $d(p,q) \leq 1$.
2. For every region $R \in \mathcal{R}$, and for every nonnegative integer h, there are at most $f(h)$ regions whose hop-distance from R in $G_{\mathcal{R},r}$ is at most h.

It is easy to see that such partitions exist:

Lemma A.1. *There exists a partition $\mathcal{R}$ and a positive real constant c_1 such that, for any $r \geq 1$, $(\mathcal{R}, r)$ is f-bounded, where $f(h) = c_1 r^2 h^2$.*

Proof. We can use a uniform grid based on squares of side $\frac{1}{2}$. To ensure that we have a partition, each square includes only some of its boundary points: the upper left corner, the upper edge excluding the endpoints, and the left edge excluding the endpoints. □

There are many other possible partitions. For the rest of the paper, we fix some partition $\mathcal{R}$ and constant c_1 satisfying the properties in Lemma A.1. We also write c_r for $c_1 r^2$; thus, c_r is a "constant" once we fix r (as we will do in much of the paper). Thus, we have:

Lemma A.2. *For the fixed partition $\mathcal{R}$ and constant c_1, for any $r \geq 1$, $(\mathcal{R}, r)$ is f-bounded, where $f(h) = c_r h^2$. In particular, any region $R \in \mathcal{R}$ has at most $c_r - 1$ neighboring regions in $G_{\mathcal{R},r}$.*

Here is another small observation relating Δ and Δ' for geographic dual graphs.

Lemma A.3. *Suppose (G, G') is an r-geographic dual graph with respect to emb, where $G = (V, E)$ and $G' = (V, E')$. Then $\Delta' \leq c_r \Delta$.*

Proof. Consider our fixed f-bounded region partition. Given our assumptions on the size of these regions, all nodes in a given region are G neighbors, and therefore the number of vertices in each region is at most Δ. Applying Lemma A.2, we get the total number of G'-neighbors, plus u itself, is at most $c_r\Delta$. □

A.2 Probability

Here are some basic probability results that we use.

Theorem A.4 (Chernoff bounds). *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \le i \le k$, $X_i \in \{0, 1\}$. Let $X = X_1 + X_2 + \cdots + X_k$ and let $\mu = \mathbb{E}[X]$. Then, for any δ, $0 \le \delta \le 1$:*

$$P[X > (1+\delta)\mu] \le e^{-\delta^2 \frac{\mu}{2}}, \tag{1}$$

$$P[X < (1-\delta)\mu] \le e^{-\delta^2 \frac{\mu}{3}}, \text{ and} \tag{2}$$

$$P[|X - \mu| < \delta\mu] \le 2e^{-\delta^2 \frac{\mu}{3}}. \tag{3}$$

We also use a standard inequality:

Lemma A.5. *Let x be a real number, $x \ge 2$. Then $(1 - \frac{1}{x})^x \ge (1/4)$.*

B Seed Agreement Analysis

Here we provide the details of our analysis of the seed agreement algorithm $SeedAlg$, defined in Section 3. In the following, recall that we have fixed a configuration, including a dual graph network topolgy: (G, G'). We have also already assumed that r is a fixed real, $r \ge 1$. We have already fixed partition $\mathcal{R}$ of the plane; now name the equivalence classes as region x, or R_x, for $x \in X$. We will say that a node u is in region x if $emb(u) \in R_x$. In a slight abuse of notation, we will sometimes use $u \in R_x$ to indicate node u is in region R_x, and $|R_x|$ to indicate the number of unique nodes in R_x.

B.1 Constants

The algorithm and the analysis use many positive real-valued constants. For reference, we collect them here:

- c_1 and $c_r = c_1 r^2$, used in specifying the region partition $\mathcal{R}$.
- $c_2 \ge 4$, in the definition of *good*, below. Let $c_3 = \frac{5}{4}c_2$.
- $c_4 \ge 2 \cdot 4^{c_r c_3}$, a parameter of the algorithm. Let $c_5 = (\frac{\log_2(e)}{12})c_4$.
- $c_6 = (\frac{1}{4})^{c_1 c_3}$.
- $\epsilon_1 \le \frac{1}{2}$, another parameter of the algorithm.
- $\epsilon_2 = (\epsilon_1)^{c_2 \log_2(e)/32} + (\epsilon_1)^{c_2 \log_2(e)/24}$, arising from some Chernoff bounds.
- $\epsilon_3 = (\epsilon_1)^{c_5 (c_6)^{r^2}}$. This high dependence on r arises in a rather delicate local analysis of success probabilities for transmission; see Lemma B.7.
- $\epsilon_4 = c_r \epsilon_2 + \epsilon_3$.

B.2 Non-Probabilistic Definitions and Lemmas

The algorithm has several interesting non-probabilistic properties. The first lemma gives some easy properties of the owners and seeds that nodes choose.

Lemma B.1. *Let α be an execution, R_x a region. Suppose that $decide(j, s)_v$ occurs in α for some node v whose region is within one hop of R_x. Suppose further that this decision is not the result of a default choice made at the end of all the phases. Then:*

1. *j is the id of some node in a region that is within two hops of R_x,*

2. *s is the initial seed chosen by j in α.*

3. *j is a leader in the phase at which the $decide(j, s)_v$ occurs.*

Proof. Straightforward. □

The remaining lemmas require some definitions. These definitions are for a particular execution α of the algorithm.

For any region (index) x and phase h, we define:

- $A_{x,h}$, $h \geq 1$, to be the set of nodes in R_x that are active at the beginning of phase h, and $a_{x,h} = |A_{x,h}|$.
- $p_h = 2^{-(\log \Delta - h+1)}$ to be the leader election probability associated with phase h, and $P_{x,h} = a_{x,h} p_h$ to be the cumulative leader election probability for R_x at phase h.
- Region x is *good* in phase h provided that $P_{x,h} \leq c_2 \log(\frac{1}{\epsilon_1})$.
- $L_{x,h}$, $h \geq 1$ to be the set of nodes in R_x that become leaders in phase h, and $\ell_{x,h} = |L_{x,h}|$.

Note that all of these notions except for $L_{x,h}$ and $\ell_{x,h}$ depend on just the first $h-1$ phases of the implicitly-assumed execution α. These last two notions depend on the first $h-1$ phases plus the initial step of phase h at which the nodes decide whether to be leaders at phase h.

Lemma B.2. *In every execution α, every region is good in Phase 1.*

Proof. Fix any region x. By definition of the algorithm, $p_1 = \frac{1}{\Delta}$. We have that $a_{x,1} \leq |R_x|$, and $|R_x| \leq \Delta$ by the first property of region partitions. Therefore, $P_{x,1} \leq \frac{\Delta}{\Delta} = 1$, which suffices. □

Lemma B.3. *Let α be an execution. Let x be a region, h a phase number. Suppose that, in α, $P_{x,h} \leq \frac{c_2}{2} \log(\frac{1}{\epsilon_1})$. Then in α, region x is good in phase $h+1$, that is, $P_{x,h+1} \leq c_2 \log(\frac{1}{\epsilon_1})$.*

Proof. We know that $a_{x,h+1} \leq a_{x,h}$ because the number of active nodes never increases from one phase to the next. Also, $p_{h+1} = 2p_h$. So $P_{x,h+1} = a_{x,h+1} p_{h+1} \leq 2a_{x,h} p_h = 2P_{x,h}$. By assumption, $2P_{x,h} \leq c_2 \log(\frac{1}{\epsilon_1})$. So $P_{x,h+1} \leq c_2 \log(\frac{1}{\epsilon_1})$, as needed. □

The next lemma constrains the total number of leaders ever elected in a region x, in terms of the $\ell_{x,h}$ values.

Lemma B.4. *Let α be an execution, x a region, ℓ a positive integer. Suppose that, in every phase h, $1 \leq h \leq \log \Delta$, the following hold:*

1. *$\ell_{x,h} \leq \ell$.*

2. *If $\ell_{x,h} \geq 1$ then after round h, all nodes in region x are inactive.*

Then the total number of leaders elected in region x in α, that is, $\Sigma_h \ell_{x,h}$, is at most ℓ.

Proof. Straightforward. □

The final non-probabilistic lemma constrains the number of nodes that decide on their own ids by default, at the end of phase $\log \Delta$.

Lemma B.5. *Let x be a region. Let α be a finite execution that completes exactly $\log \Delta - 1$ phases. Suppose that, in α, region x is good at phase $\log \Delta$. Then in all extensions of α, the number of nodes in region x that decide on their own ids by default, at the end of phase $\log \Delta$, is at most $2c_2 \log(\frac{1}{\epsilon_1})$.*

Proof. Since region x is good in phase $\log \Delta$, we know that $P_{x,\log \Delta} \leq c_2 \log(\frac{1}{\epsilon_1})$. Since $p_{\log \Delta} = \frac{1}{2}$ and $P_{x,\log \Delta} = a_{x,\log \Delta} p_{\log \Delta}$, we have that $a_{x,\log \Delta} \leq 2c_2 \log(\frac{1}{\epsilon_1})$. That is, the number of nodes in region x that are active at the beginning of phase $\log \Delta$ is at most $2c_2 \log(\frac{1}{\epsilon_1})$. This bounds the number of nodes that decide on their own ids by default, at the end of phase $\log \Delta$. □

B.3 Probabilistic Lemmas about Individual Phases

In this subsection and the next, we prove some lemmas about the probabilities for having certain numbers of leaders, for successful transmission, and for "goodness". This subsection contains lemmas about behavior in a single phase. The next subsection builds upon this one to describe behavior in many phases.

B.3.1 Number of leaders

The first lemma provides bounds on the number of leaders that are chosen in a region x at a phase h, based on the range of values for the probability sum $P_{x,h}$.

Lemma B.6. *Let x be a region, h a phase number. Let α be a finite execution that completes exactly $h - 1$ phases. (Note that this is enough to determine $P_{x,h}$.) Then, considering probabilities in extensions of α:*

1. *If $P_{x,h} \leq c_2 \log(\frac{1}{\epsilon_1})$, then with probability at least $1 - (\epsilon_1)^{c_2 \log_2(e)/32}$, $\ell_{x,h} \leq \frac{5}{4} c_2 \log(\frac{1}{\epsilon_1})$.*
2. *If $\frac{c_2}{2} \log(\frac{1}{\epsilon_1}) \leq P_{x,h}$ then with probability at least $1 - (\epsilon_1)^{c_2 \log_2(e)/24}$, $\ell_{x,h} \geq \frac{1}{4} c_2 \log(\frac{1}{\epsilon_1})$.*

Proof. Define an indicator variable $\chi(u)$ for each $u \in A_{x,h}$ (i.e., the set of active nodes in R_x at the beginning of phase h): $\chi(u) = 1$ if u elects itself leader in this phase (i.e., $u \in L_{x,h}$), and $\chi(u) = 0$ otherwise (i.e., $u \neq L_{x,h}$). We express the total number of leaders in this region and phase as a sum of these indicators: $\ell_{x,h} = \sum_{u \in A_{x,h}} \chi(u)$. For each u, we have that $\mathbb{E}[\chi(u)] = p_h$. By linearity of expectation, $\mathbb{E}[\ell_{x,h}] = a_{x,h} p_h = P_{x,h}$.

1. We upper bound the probability that $\ell_{x,h} > \frac{5}{4} c_2 \log(\frac{1}{\epsilon_1})$ using a Chernoff bound. Let $\mu = P_{x,h}$. We define δ so that $(1 + \delta)\mu = \frac{5}{4} c_2 \log(\frac{1}{\epsilon_1})$, specifically, let $\delta = (\frac{5}{4} c_2 \log(\frac{1}{\epsilon_1}))/P_{x,h} - 1$. Then Chernoff yields a probability upper bound of $e^{-\delta^2 \mu/2}$. If we express this as a function of $P_{x,h}$, we see that this bound increases monotonically as $P_{x,h}$ increases. So the largest value of the expression occurs when $P_{x,h} = c_2 \log(\frac{1}{\epsilon_1})$. The expression then works out to $e^{-(1/4)^2 (1/2) c_2 \log(\frac{1}{\epsilon_1})}$, which is equal to $e^{-1/32 c_2 \log(\frac{1}{\epsilon_1})} = (\epsilon_1)^{c_2 \log_2(e)/32}$. Then the probability that $\ell_{x,h} \leq \frac{5}{4} c_2 \log(\frac{1}{\epsilon_1})$ is at least $1 - (\epsilon_1)^{c_2 \log_2(e)/32}$.

2. We upper bound the probability that $\ell_{x,h} < \frac{1}{4}c_2 \log(\frac{1}{\epsilon_1}) = \frac{1}{2}(c_2/2)\log(\frac{1}{\epsilon_1})$, using Chernoff. To apply Chernoff, we use $\mu = P_{x,h}$. We define δ so that $(1-\delta)\mu = \frac{1}{4}c_2\log(\frac{1}{\epsilon_1})$. This works out to $\delta = 1 - (\frac{1}{4}c_2\log(\frac{1}{\epsilon_1}))/P_{x,h}$. Then Chernoff yields a probability upper bound of $e^{-\delta^2\mu/3}$. If we express this as a function of $P_{x,h}$ we see that this bound decreases monotonically as $P_{x,h}$ increases. So the largest value of the expression occurs when $P_{x,h} = (c_2/2)\log(\frac{1}{\epsilon_1})$. The expression then works out to $e^{-(1/2)^2(1/3)(c_2/2)\log\frac{1}{\epsilon_1}}$, which is equal to $e^{-1/24 c_2 \log(\frac{1}{\epsilon_1})} = (\epsilon_1)^{c_2\log_2(e)/24}$. Then the probability that $\ell_{x,h} \geq \frac{1}{4}c_2\log(\frac{1}{\epsilon_1})$ is at least $1-(\epsilon_1)^{c_2\log_2(e)/24}$.

□

B.3.2 Successful transmission

The next lemma provides a bound for successful transmission in a region x at a phase h, based on bounds on the number of leaders chosen at phase h in region x and its neighboring regions.

Lemma B.7. *Let x be a region, h a phase number. Let α be a finite execution that completes exactly $h-1$ phases, plus the leader election step of phase h. Suppose that, in α:*

1. *Region x satisfies $1 \leq \ell_{x,h} \leq c_3\log(\frac{1}{\epsilon_1})$.*
2. *Every neighboring region y of x satisfies $\ell_{y,h} \leq c_3\log(\frac{1}{\epsilon_1})$.*

Let u be any element of $L_{x,h}$. Then with probability at least $1-\epsilon_3$ in extensions of α, there is some round in phase h in which u transmits and no other node in R_x or node in any neighboring region R_y transmits.

Proof. First, we lower bound the probability that this combination of events occurs in any single round of phase h by:

$$(1/\log(\frac{1}{\epsilon_1}))(1-1/\log(\frac{1}{\epsilon_1}))^{c_r c_3 \log(\frac{1}{\epsilon_1})};$$

This is because u transmits with probability $1/\log(\frac{1}{\epsilon_1})$ and there are at most $c_r c_3\log(\frac{1}{\epsilon_1})$ transmitting nodes in region R_x plus its neighboring regions. By Lemma A.5, the right-hand side of this inequality is at least

$$(1/\log(\frac{1}{\epsilon_1}))(\frac{1}{4})^{c_r c_3}.$$

Now consider all of phase h. Since the individual round probabilities are at least $(1/\log(\frac{1}{\epsilon_1}))(1/4)^{c_r c_3}$, and we have $c_4\log^2(\frac{1}{\epsilon_1})$ rounds, the mean number of successes is at least the product of these two expressions, which is $c_4\log(\frac{1}{\epsilon_1})(1/4)^{c_r c_3}$. Now we take $\delta = \frac{1}{2}$ in the Chernoff lower bound, Theorem A.4, Equation 2 and get that we have at least half the mean number of successes, hence at least one success, with probability at least $1 - e^{-(1/12)c_4\log(\frac{1}{\epsilon_1})(\frac{1}{4})^{c_r c_3}}$. (Here we use the assumed lower bound on c_4.) This simplifies to: $1-(\epsilon_1)^{(1/12)c_4\log_2(e)(\frac{1}{4})^{c_r c_3}} = 1-(\epsilon_1)^{c_5(c_6)^{r^2}} = 1-\epsilon_3$. □

Note that the error bound ϵ_3 that appears in Lemma B.7 has a double-exponential dependence on r. We do not know how to avoid this. To compensate for large r, we would need to use small values of ϵ_1, which would impact the running time of the algorithm. This suggests that, for this approach to be feasible in practice, one would need to have small values of r.

B.3.3 Goodness

We prove a lemma that says how goodness is preserved for a region R_x at one phase h. It turns out that preserving goodness for region x at phase h depends on goodness of all neighboring regions of x at the beginning of phase h, but not on goodness of any regions that are further away. The analysis is more delicate than corresponding arguments in [11] because we are seeking local bounds.

Lemma B.8. *Let x be any region, h a phase number. Let α be a finite execution that completes exactly $h-1$ phases. Suppose that, in α, region x and all its neighboring regions in the region graph are good at phase h. Then with probability at least $1-\epsilon_4$ in extensions of α, region x is also good at phase $h+1$.*

Proof. By assumption, $P_{x,h} \leq c_2 \log(\frac{1}{\epsilon_1})$. If $P_{x,h} \leq \frac{c_2}{2} \log(\frac{1}{\epsilon_1})$, that is, if region x is far below the threshold used to define *good*, then Lemma B.3 implies that region x is good in phase $h+1$ (in all extensions), which suffices. So for the remainder of the proof, we assume that $P_{x,h} > \frac{c_2}{2} \log(\frac{1}{\epsilon_1})$, that is, region x is fairly close to the threshold.

Lemma B.6 implies that, with probability at least $1-\epsilon_2$, $\frac{1}{4}c_2 \log(\frac{1}{\epsilon_1}) \leq \ell_{x,h} \leq \frac{5}{4}c_2 \log(\frac{1}{\epsilon_1})$. This implies that, with probability at least $1-\epsilon_2$, $1 \leq \ell_{x,h} \leq c_3 \log(\frac{1}{\epsilon_1})$. (We use the lower bound on c_2 here.) Likewise, for any neighboring region y of x, Lemma B.6 implies that, with probability at least $1-\epsilon_2$, $\ell_{y,h} \leq c_3 \log(\frac{1}{\epsilon_1})$. Since the number of regions within one hop of x is at most c_r, we can use a union bound to conclude that with probability at least $1-c_r(\epsilon_2)$, both of the following hold: (1) $1 \leq \ell_{x,h} \leq c_3 \log(\frac{1}{\epsilon_1})$, and (2) for every region y that is a neighbor of x, $\ell_{y,h} \leq c_3 \log(\frac{1}{\epsilon_1})$. That is, with "high probability", no region within one hop of x elects more than $c_3 \log(\frac{1}{\epsilon_1})$ leaders, and x elects at least one leader.

We argue that this combination of constraints on the P values is well suited for region x to succeed in broadcasting to the region and reduce the sum to 0 by the start of phase $h+1$. Formally, let B be the set of executions that extend α for just the initial leader-election step of phase h, and choose leaders at phase h in such a way that conditions (1) and (2) above are satisfied. Then for each particular $\alpha' \in B$, fix $u_{\alpha'} \in L_{x,h}$ to be any one of the leaders that are chosen in region x in phase h. Then Lemma B.7 implies that, in extensions of α', $u_{\alpha'}$ succeeds in delivering a message to all active nodes in region x with probability at least $1-\epsilon_3$. In this case, all nodes in region x start phase $h+1$ inactive.

Then by Total Probability, the probability, conditioned on B, that some leader in region x succeeds in delivering a message to all active nodes in region x, is at least $1-\epsilon_3$.

Now we use a union bound to combine the two probabilities—for selecting the "right" number of leaders and for the broadcast succeeding, and we get that all this happens with probability at least $1-c_r(\epsilon_2)-\epsilon_3 = 1-\epsilon_4$. In this case, all nodes in region x start phase $h+1$ inactive. Therefore, $P_{x,h+1} = 0$ which clearly satisfies the definition of *good*. □

Lemma B.8 gives a bound on preserving goodness in extensions of some particular $(h-1)$-phase execution α. The following corollary gives a similar bound for all executions taken together.

Corollary B.9. *Let x be any region. Let h be a phase number. Suppose (condition on the event) that region x and all its neighboring regions in the region graph are good at phase h. Then with probability at least $1-\epsilon_4$, region x is also good in phase $h+1$.*

Proof. This follows from Lemma B.8, using Total Probability (considering all prefixes α satisfying the assumption). □

B.4 Probabilistic Lemmas about Multiple Phases

We now use the results of the previous section to get results about what happens during many phases of an execution. We begin with a lemma that gives conditions for a region x and all its neighboring regions to a designated distance to remain good for phases $1, \ldots, n$. This proof relies on Corollary B.9 for the inductive step. The lemma is stated quite generally, although later, in the analysis of $SeedAlg$, we will need the result for only limited distances.

Lemma B.10. *Let x be any region, h and k integers, $1 \leq h \leq \log \Delta$, $0 \leq k$. Then x and all its neighboring regions to distance $\log \Delta + k - h$ are good at phases $1, \ldots, h$, with probability at least $1 - c_r(h-1)(\log(\Delta) + k)^2 \epsilon_4$.*

Proof. By induction on h, for each fixed k. The base, $h = 1$, follows from Lemma B.2. For the inductive step, assume the lemma for h and prove it for $h+1$, where $1 \leq h \leq \log \Delta - 1$. That is, assume that region x and all its neighboring regions to distance $\log \Delta + k - h$ are good at phases $1, \ldots, h$, with probability at least $1 - c_r(h-1)(\log(\Delta) + k)^2 \epsilon_4$. We show that x and all its neighboring regions to distance $\log \Delta + k - (h+1)$ are good at phases $1, \ldots, h+1$, with probability at least $1 - c_r h(\log(\Delta) + k)^2 \epsilon_4$. Let A be the set of executions in which all regions within $\log \Delta + k - h$ hops of region x are good at phases $1, \ldots, h$; thus, $Pr(A) \geq 1 - c_r(h-1)(\log(\Delta) + k)^2 \epsilon_4$.

Now consider any particular region y that is within $\log \Delta + k - (h+1)$ hops of x. Then y and all its neighboring regions are within $\log \Delta + k - h$ hops of x, and so, are good at phases $1, \ldots, h$ in every execution in A. Corollary B.9 implies that, with probability at least $1 - \epsilon_4$, conditioned on A, region y is also good at phase $h+1$. There are at most $c_r(\log(\Delta) + k)^2$ such regions y, by definition of the region partition. So, with probability at least $1 - c_r(\log(\Delta) + k)^2 \epsilon_4$, conditioned on A, all such y are good at phase $h+1$. Thus, with probability at least $1 - c_r(\log(\Delta) + k)^2 \epsilon_4$, conditioned on A, all such y are good at phases $1, \ldots, h+1$.

Combining this probability bound with the probability bound for A, we conclude that with probability at least $1 - c_r(h-1)(\log(\Delta) + k)^2 \epsilon_4 - c_r(\log(\Delta) + k)^2 \epsilon_4 = 1 - c_r h(\log(\Delta) + k)^2 \epsilon_4$, all such y are good at phases $1, \ldots, h+1$. □

Corollary B.11. *With probability at least $1 - c_r(\log(\Delta))(\log(\Delta) + k)^2 \epsilon_4$, region x and all its neighboring regions to distance k are good in every phase h, $1 \leq h \leq \log \Delta$.*

Proof. Apply Lemma B.10 for $h = \log \Delta$. □

Next, we define some special events (sets of executions), based on goodness, bounded numbers of leaders, and transmission success, and prove bounds for these. We will use the final lemma, about transmission success, in the final analysis of $SeedAlg$.

For $1 \leq h \leq \log \Delta$, define G_h to be the set of executions in which every region within three hops of region x is good at all phases $1, \ldots, h$.

Lemma B.12. $Pr(G_{\log(\Delta)}) \geq 1 - c_r(\log(\Delta) + 3)^3 \epsilon_4$.

Proof. By Corollary B.11. □

For $1 \leq h \leq \log \Delta$, define L_h to be the set of executions in which every region y within three hops of region x satisfies the condition $\ell_{y,h} \leq c_3 \log(\frac{1}{\epsilon_1})$.

Lemma B.13. *For any phase h, $Pr(L_h) \geq 1 - c_r(\log(\Delta) + 3)^3 \epsilon_4 - 9c_r\epsilon_2$.*

Proof. We have that $Pr(L_h) \geq Pr(G_h \cap L_h) = Pr(G_h)Pr(L_h|G_h)$. We know that $Pr(G_h) \geq Pr(G_{\log(\Delta)}) \geq 1 - c_r(\log(\Delta)+3)^3 \epsilon_4$, by Lemma B.12. By Lemma B.6, we have that $Pr(L_h|G_h) \geq 1 - 9c_r(\epsilon_1)^{c_2 \log_2(e)/32} \geq 1 - 9c_r\epsilon_2$, using the fact that there are at most $9c_r$ regions within three hops of x. Combining these inequalities, we get that $Pr(L_h) \geq 1 - c_r(\log(\Delta) + 3)^3 \epsilon_4 - 9c_r\epsilon_2$. □

For $1 \leq h \leq \log \Delta$, define S_h to be the set of executions in which every region y within two hops of region x satisfies the following property: "If $\ell_{y,h} \geq 1$ then there is some round in phase h in which some node u in region y transmits and no other node in region y or any neighboring region z transmits." Write S as shorthand for $S_1 \cap \ldots \cap S_{\log \Delta}$, i.e., S is the set of executions in which all regions within two hops of x "have successful transmissions, if possible" at all phases.

Lemma B.14. $Pr(S_h) \geq 1 - c_r(\log(\Delta) + 3)^3 \epsilon_4 - 9c_r\epsilon_2 - 4c_r\epsilon_3$.

Proof. We have that $Pr(S_h) \geq Pr(L_h \cap S_h) = Pr(L_h)Pr(S_h|L_h)$. By Lemma B.13, we know that $Pr(L_h) \geq 1 - c_r(\log(\Delta) + 3)^3\epsilon_4 - 9c_r\epsilon_2$. By Lemma B.7, we have that $Pr(S_h|L_h) \geq 1 - 4c_r\epsilon_3$, using the fact that there are at most $4c_r$ regions within two hops of x. Combining these inequalities, we get that $Pr(S_h) \geq 1 - c_r(\log(\Delta) + 3)^3\epsilon_4 - 9c_r\epsilon_2 - 4c_r\epsilon_3$. □

Lemma B.15. $Pr(S) \geq 1 - c_r(\log \Delta)[(\log(\Delta) + 3)^3\epsilon_4 + 9\epsilon_2 + 4\epsilon_3]$.

Proof. By Lemma B.14. □

B.5 Correctness of $SeedAlg$

Now we use the lemmas in the previous subsections to show that our algorithm satisfies the seed-service specification. Our overall goal is to prove:

Theorem B.16. *$SeedAlg(\epsilon_1)$ satisfies the $Seed(\delta, \epsilon)$ specification, where δ is $O(r^2 \log(\frac{1}{\epsilon_1}))$, and $\epsilon = O(r^4 \log^4(\Delta)(\epsilon_1)^{c^{r^2}})$, where c is some constant, $0 < c < 1$. The algorithm takes $O((\log \Delta) \log^2(\frac{1}{\epsilon_1}))$ rounds.*

Proof. The time complexity is immediate from the definition of the algorithm. It remains to show that the algorithm meets the specification. Properties 1 and 2 are straightforward. Property 4 should be easy because the choices are made independently. It remains to show Property 3, the δ bound on the number of local seeds.

For Property 3, fix some node u, in some region x. We prove a "high probability" bound on the number of different owners/seeds that are decided upon within one hop of region x. There are two ways a node can decide on a seed: (a) "Normally", either by using its own seed because it elects itself a leader, or because it adopts the seed of another leader. (b) "By default", because it never elects itself leader or receives a seed, so it adopts its own seed at the end of the execution. We bound the number of owners/seeds of these two types separately.

First we bound the number of normal decisions. Let α be any execution in S. Then by the way the algorithm works, for any region y that is within two hops of x, and for any phase h, the following condition holds: If there is some round of phase h in which some node in region y transmits and no other node in region y or any neighboring region transmits, then all nodes in region y are inactive in phases $> h$. Then Lemma B.4 implies that, throughout α, region y elects at most $c_3 \log(\frac{1}{\epsilon_1})$ leaders. By our geographic constraint, the number of such regions y is at most $4c_r$, so the total number of leaders elected in regions that are within two hops of x is at most $4c_rc_3 \log(\frac{1}{\epsilon_1})$. Therefore, in α, the total number of unique seeds that are decided upon "normally" by nodes in regions that are within one hop of x is also at most $4c_rc_3 \log(\frac{1}{\epsilon_1})$. Therefore, by Lemma B.15, with probability at least $1 - c_r(\log \Delta)[(\log(\Delta) + 3)^3\epsilon_4 + 9\epsilon_2 + 4\epsilon_3]$, the total number of unique seeds that are decided upon "normally" by nodes within one hop of x is at most $4c_rc_3 \log(\frac{1}{\epsilon_1})$.

Next, we bound the number of default decisions. Consider any execution α in $G_{\log(\Delta)}$. Lemma B.5 implies that, in α, in any region y that is within one hop of region x, at most $2c_2 \log(\frac{1}{\epsilon_1})$ nodes decide by default. So the total number of nodes in all such regions y that decide by default in α is at most $2c_r c_2 \log(\frac{1}{\epsilon_1})$. Therefore, by Lemma B.12, with probability at least $1-c_r(\log(\Delta)+3)^3\epsilon_4$, the total number of unique seeds that are decided upon "by default" by nodes within one hop of x is at most $2c_r c_2 \log(\frac{1}{\epsilon_1})$.

Combining the results for normal and default decisions using another union bound, we get that the total number of different owner/seeds that are decided upon within one hop of x are at most

$$4c_r c_3 \log(\frac{1}{\epsilon_1}) + 2c_r c_2 \log(\frac{1}{\epsilon_1}) \leq 6c_r c_3 \log(\frac{1}{\epsilon_1}),$$

with probability at least $1 - c_r(\log\Delta)[(\log(\Delta)+3)^3\epsilon_4 + 9\epsilon_2 + 4\epsilon_3] - c_r(\log(\Delta)+3)^3\epsilon_4$ □

B.6 Useful Seed Properties

The two lemmas below describe useful properties of the seeds committed by $SeedAlg$, they are defined with respect to the execution distribution resulting from combining $SeedAlg$ with a configuration. The first of these lemmas follows from a straightforward application of the *independence* property of the $Seed$ specification implemented by $SeedAlg$.

Lemma B.17. *Fix some process i, integer $k, 1 \leq k < \kappa$, and bit string $q \in \{0,1\}^k$. Let s_i be the seed i commits after the $SeedAlg$ subroutine completes. It follows that* $\Pr(s_i[k+1] = 0 \mid s_i[1]s_i[2]...s_i[k] = q) = \Pr(s_i[k+1] = 1 \mid s_i[1]s_i[2]...s_i[k] = q) = \frac{1}{2}$.

Whereas the above lemma captured that independence of each bit in a seed from its previous bits, it is also important to establish independence between *different* seeds. The following lemma accomplishes this goal. It too can be established as straightforward application of the *independence* property:

Lemma B.18. *Fix two processes i and j and let s_i and s_j be the seeds committed by i and j, respectively. Assume s_i and s_j have different seed owners. It follows that these seed values are independent.*

C Local Broadcast Analysis

Here we contain the details of our analysis of the local broadcast algorithm $LBAlg$ described in Section 4.

C.1 Constants

Below is a summary of the constants used in algorithm description and analysis that follow. Notice, we reuse some constant names (e.g., c_1, c_2, c_3...) also used in Section 3. These new definitions override the old definitions. In the following, we assume Δ is a power of two (to reduce notation).

- ϵ_1 is the notation used in the below algorithm description to describe the desired error probability.
- ϵ' is the maximum error probability bound that guarantees, given the constraints of Theorem 3.1, that $SeedAlg(\epsilon')$ satisfies the $Seed(\delta, \epsilon)$ spec for an $\epsilon \leq \epsilon_1/2$.

  Note: given the relationship between $SeedAlg$ and $Seed$'s error bounds, as established in Theorem 3.1, $\epsilon' = \Theta\big((\frac{\epsilon_1}{r^4 \log^4 \Delta})^{(1/(cr^2))}\big)$, where c, $0 < c < 1$, is the constant provided by Theorem 3.1. Because $c < 1$, we can rewrite the bound as $\Theta\big((\frac{\epsilon_1}{r^4 \log^4 \Delta})^{(\gamma/r^2)}\big)$, for some constant $\gamma > 1$.
- $\epsilon_2 = \min\{\epsilon', \epsilon_1\}$.

  Note: For asymptotic concision, we want to ensure that the error probability we use for $SeedAlg$ is no more than ϵ_1. We cannot simply claim that ϵ', as defined above, satisfies this constraint because given its relationship to to ϵ_1 from above, it is possible that if γ is sufficiently small compared to r^2, and ϵ_1 is sufficiently small compared to r and $\log \Delta$, that this exponent will be a sufficiently small fraction to increase $(\frac{\epsilon_1}{r^4 \log^4 \Delta})$ to something larger than ϵ_1. This $\min$ statement handles this possibility.

- c_2 is a constant used in our analysis of successful receptions of messages.
- c_1 is a constant we use in defining the length of a phase in the algorithm (see T_{prog} below).
- $T_{prog} = \lceil c_1 \cdot r^2 \cdot \log(\frac{1}{\epsilon_1}) \cdot \log(\frac{1}{\epsilon_2}) \cdot \log \Delta \rceil = O\big(r^2 \log(\frac{1}{\epsilon_1}) \log(\frac{1}{\epsilon_2}) \log \Delta\big)$ is the number of rounds required by our algorithm to ensure progress.
- $\kappa = T_{prog} \cdot \lceil \log(r^2 \log(\frac{1}{\epsilon_2})) \rceil \cdot \log\log \Delta$: the maximum number of bits consumed from a seed agreement seed in a single *phase* of length T_{prog} worth of broadcasting.
- $T_{ack} = \frac{r^2 12 \log(1/\epsilon_2) \log \Delta \ln(\frac{2\Delta}{\epsilon_1})\Delta'}{c_2 T_{prog}(1-\epsilon_1/2)} = \frac{12 \ln(\frac{2\Delta}{\epsilon_1})\Delta'}{c_2 c_1 \log(1/\epsilon_1)(1-\epsilon_1/2)} = O\big(\frac{\Delta \log(\Delta/\epsilon_1)}{(1-\epsilon_1)}\big)$ describes the number of phases a node will spend attempting to send a message that arrives as a $bcast$ input.
- Let $T_s = O(\log \Delta \log^2(\frac{1}{\epsilon_2}))$ be the number of rounds required for the seed agreement algorithm $SeedAlg(\epsilon_2)$ (as provided by Theorem 3.1).

C.2 Analysis

Our goal in this section is to prove Theorem 4.1: the main correctness theorem for $LBAlg$, described in Section 4.3. To begin, fix a configuration consisting of a dual graph $(G = (V, E), G' = (V, E'))$, process assignment, link scheduler, and allowable environment. Also fix an allowable error probability ϵ_1. Notice, this configuration combined with the $LBAlg(\epsilon_1)$ algorithm specifies a distribution over executions, which we can describe as an execution tree. The remainder of this argument concerns this distribution.

We now bound the behavior of $LBAlg$ within the scope of a single phase. To do so, we first introduce some useful notation. For a given phase i of an execution, let B_i be the set of nodes that are in sending status during phase i, and $R_i = N_G(B_i)$ be the set of nodes that neighbor B_i in G. Notice, because sending status is fixed for the duration of a given phase, both B_i and R_i are determined at the beginning of phase i and cannot change during the phase. Using this notation, we specify and prove the following key probabilistic behavior:

Lemma C.1. *Fix some phase i and an execution prefix through the $(j-1)^{th}$ body round of this phase, for some $j \in \{2, ..., T_{prog}\}$. Fix nodes u and v, where $u \in R_i$ and $v \in N_G(u) \cap B_i$. Assume the call to $SeedAlg$ at the beginning of phase i in this prefix satisfies $B_{u,\delta}$. Let p_u be the probability that u receives some message in the j^{th} round, and let $p_{u,v}$ be the probability that u receives a message* from v *in this round. It follows that:*

- $p_u \geq \frac{c_2}{r^2 \log(\frac{1}{\epsilon_2}) \log \Delta}$
- $p_{u,v} \geq p_u / \Delta'$

Proof. Fix some u, v, t and a prefix, as specified by the lemma statement. (Notice, we know that a node v satisfying the constraints of the statement exists due to the assumption that $u \in R_i$, which implies that $|N_G(u) \cap B_i| > 0$.) Let $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ be a minimum-sized partition of the nodes in $N_{G'}(u) \cap B_i$ such that all nodes in S_j committed to the same seed in the beginning of this phase. Given the lemma assumption that the preamble of this phase satisfies $B_{u,\delta}$, it follows: $k \leq \delta$. Finally, let S_{i_v} be the set from $\mathcal{S}$ that contains v.

We now analyze the next broadcast round. In this round, nodes in B_i use their seeds to decide whether or not to become a participant. In particular, they become a participant with probability $\frac{1}{r^2 \log(\frac{1}{\epsilon_2})} = c/\delta$, for some constant $c > 0$, using bits from their seeds to resolve the random choice. For each $S_j \in \mathcal{S}$, all nodes in S_j make the same decision in each round because they are using bits from the same seed. Let p_{i_v} be the probability that set S_{i_v} decides to be a participant, and *all other* sets in $\mathcal{S}$ decide to be non-participants. To bound p_{i_v}, we apply Lemmata B.17 and B.18 to obtain the uniformity and independence properties needed to prove the following:

$$\begin{aligned} p_{i_v} &= (c/\delta)(1-(c/\delta))^{k-1} \\ &> (c/\delta)(1/4)^{\frac{c(k-1)}{\delta}} \\ &\overset{(k\leq\delta)}{>} (c/\delta)(1/4)^{c} \\ &= \Theta(1/\delta) \end{aligned}$$

Assume this event—that only S_{i_v} decides to participate from among the sets in $\mathcal{S}$—occurs. It follows that only nodes in S_{i_v} can potentially broadcast in this round. Let ℓ be the number of links from nodes in S_{i_v} to u included in the network topology for the round by the link scheduler included in our configuration definition. Because S_{i_v} contains the G-neighbor v (by definition), we know that v is connected to u and that therefore $\ell > 0$.

The next thing that happens in this round is that the nodes in S_{i_v} use more random bits from their shared seed to choose a value uniformly from $[\log \Delta]$. If $\ell = 1$, we define the *correct* choice of value from $[\log \Delta]$ to be 1. If $\ell > 1$, we define the *correct* choice to be $\lceil \log \ell \rceil$. By Lemma B.17, we know the nodes in S_{i_v} will choose a value from this set with uniform probability. The probability they choose a correct value with respect to ℓ is therefore at least $1/\log \Delta$.

Assume that this event also occurs. At this point, by assumption, only nodes in S_{i_v} are potential broadcasters. Each such node decides to broadcast with the correct probability, p_c, which, as defined above, is within a factor of 2 of $1/\ell$. Let us consider the possibilities. We first note that with probability at least $1/2$, u will decide to receive in this round (broadcast probability $1/2$, which corresponds to choosing the value 1 from $[\log \Delta]$, is the largest possible broadcast probability).

Assume this event occurs. The probability that *exactly one* neighbor among the ℓ neighbors connected to u subsequently decides to broadcast is constant (as there are ℓ neighbors, each deciding to broadcast with probability $p_c = \Theta(1/\ell)$). To calculate p_u we must now combine all three independent probabilities: the $\Theta(1/\delta)$ probability that S_{i_v} is the only set in $\mathcal{S}$ to participate, the $1/\log \Delta$ probability that S_{i_v} nodes choose the correct value, and the $\Theta(1)$ probability that u decides to receive and exactly one neighbor of u in the topology for the round broadcasts. We combine the constants in these asymptotic expression to define a lower bound on the constant c_2 used in our definition of p_u from the lemma statement.

Now we step back to consider $p_{u,v}$. Whereas we just calculated that there is a constant probability that exactly one node from among ℓ nodes decides to broadcast using broadcast probability p_c, we must now ask the probability that a specific node—i.e., v—is this broadcaster. We can bound this probability as:

$$p_c(1-p_c)^{\ell-1} = \frac{c'}{\ell}(1-\frac{c'}{\ell})^{\ell-1} \geq \frac{c'}{\ell}(\frac{1}{4})^{c'} \geq \frac{1}{4\Delta'},$$

where c' is a constant of size at least 1 used in the definition of p_c. By replacing the constant probability for this final step used in the derivation of p_u above with this new $\approx 1/\Delta'$ probability, we get the $p_{u,v}$ bound required by the lemma statement. (As a slight technicality, we omit the $1/4$ in the $\frac{1}{4\Delta'}$ calculation above in this final $p_{u,v}$ bound, as this can be captured by adjusting the constant c_2 calculated for p_u by a factor of 4 to include this extra amount.) □

We can now draw on Lemma C.1 to prove the progress and reliability properties required by the LB specification. We begin with progress:

Lemma C.2. *$LBAlg(\epsilon_1)$ solves the $LB(t_{ack}, t_{prog}, \epsilon_1)$ problem for: $t_{prog} = T_s + T_{prog}$.*

Proof. Notice that this definition of t_{prog} is the same as the length used by the phases in our algorithm. It follows that the boundaries of the phases in the progress property align with the phase boundaries used by $LBAlg$, so we can refer to both types of phases interchangeably. To prove progress, therefore, it is sufficient to show that for any node u and phase i such that u has an active G neighbor, the probability that u receives at least one broadcast message in this phase is at least $1 - \epsilon_1$.

To do so, fix some node u, phase i, and phase i prefix α such that A^u_α is non-empty; i.e., there is at least one G neighbor of u that is active throughout phase i. Let α' be the extension of α through the call to $SeedAlg$ at the beginning of phase i. By Theorem 4.16, and the definition of the ϵ_2 error parameter passed to $SeedAlg$, this call to $SeedAlg$ in this α' extension satisfies $B_{u,\delta}$ with probability at least $1 - (\epsilon_1/2)$.

Assume our above assumptions (including the assumption that $B_{u,\delta}$ is satisfied) hold. It follows that Lemma C.1 applies with respect to u for all T_{prog} of the subsequent body rounds in phase i Let $p^{(j)}_u$ be the (independent) probability that u receives a message in body round j of the phase. Notice, that Lemma C.1 tells us that $p^{(j)}_u \geq p_u \geq \frac{c_2}{r^2 \log(\frac{1}{\epsilon_2}) \log \Delta}$ for each such round. We can now bound the probability that u *fails* to receive a message in all T_{prog} body rounds as:

$$p_{fail} = \prod_{j=1}^{T_{prog}} (1 - p^{(j)}_u) \leq \prod_{j=1}^{T_{prog}} (1 - p_u) < (1/e)^{T_{prog} p_u} = (1/e)^{c_1 c_2 \cdot \log(1/\epsilon_1)}.$$

It is straightforward to show that for sufficiently large values of constants c_1 and c_2, we get $p_{fail} \leq \epsilon_1/2$. To conclude the proof, we use a union bound to show that the probability that $B_{u,\delta}$ does not hold and/or the probability that u fails to receive a message when this property does hold, is less than ϵ_1: providing the needed $1 - \epsilon_1$ probability for u receiving a least one message in phase i. □

We now turn our attention to the reliability property of our local broadcast problem:

Lemma C.3. *$LBAlg(\epsilon_1)$ solves the $LB(t_{ack}, t_{prog}, \epsilon_1)$ problem for: $t_{ack} = (T_{ack} + 1)(T_{prog} + T_s)$.*

Proof. Fix some nodes u and v that are neighbors in G. Let $k = \ln(\frac{2\Delta}{\epsilon_1})/p$, for $p = \frac{c_2}{r^2 \log(\frac{1}{\epsilon_2}) \log \Delta}$ (i.e., p is the lower bound for $p_{u,v}$ from the statement of Lemma C.1). In the following, we define a body round to be *useful* with respect to u, if it occurs in a phase such that $B_{u,\delta}$ holds for the preceding $SeedAlg$ preamble. Let p_1 be the probability that u fails to receive a message m from v during k useful rounds in which v is active with m. Applying Lemma C.1 to lower bound the receive probability in each of these rounds, it follows:

$$p_1 \leq (1 - p)^k < (1/e)^{pk} = \frac{\epsilon_1}{2\Delta}.$$

We now investigate the number of phases necessary to ensure that v experiences at least k useful rounds with a sufficiently high probability. To do so, we first fix $q = \lceil \frac{12k}{T_{prog}(1-\epsilon_1/2)} \rceil$. Consider an experiment where we run q consecutive phases. Let X_i, for $i \in [q]$, be a random variable that describes the number of useful rounds in phase i of the experiment. Notice, X_i either takes on the value T_{prog} (with probability at least $1 - \epsilon_1/2$) or 0 (else). Let $Y = X_1 + X_2 + ... + X_q$ be the total number of useful rounds generated by the experiment. It follows:

$$\mathbb{E}[Y] = q \cdot T_{prog} \cdot (1 - \epsilon_1/2) = 12k.$$

We now apply a Chernoff Bound (in particular, Form 2 from Theorem A.4), to bound the probability that Y is more than a factor of 2 smaller than its expectation $\mu = \mathbb{E}[Y]$:

$$\Pr(Y < (1/2)\mu = 6k) < e^{-\frac{\mu}{12}} = e^{-k} \leq e^{-\ln{(\frac{2\Delta}{\epsilon_1})}} = \epsilon_1/(2\Delta).$$

(Notice, in the above we can bound $e^{-k} \leq e^{-\ln{(\frac{2\Delta}{\epsilon_1})}}$ because $k \geq \ln{(\frac{2\Delta}{\epsilon_1})}$.)

We have now established that with probability at least $1 - \epsilon_1/(2\Delta)$, u will experience at least $6k > k$ useful rounds in q phases. We earlier established that if u experiences at least k useful rounds during which v is broadcasting m, then u receives m from v with probability at least $1 - \epsilon_1/(2\Delta)$. Assume u broadcasts m for at least q consecutive phases. By a union bound, the probability that both events occur with respect to these phases, and u therefore receives v's message m, is greater than $1 - \epsilon_1/\Delta$.

To conclude, we want to calculate, under the assumption that v broadcasts m for at least q phases, that *every* G neighbor of v succeeds in receiving m. Because there are at most Δ such neighbors, and each succeeds with probability at least $1 - \epsilon_1/\Delta$, a union bound says that every neighbor succeeds with probability at least $1 - \epsilon_1$, as required by the reliability property.

To satisfy reliability, therefore, it is sufficient for any node v receiving a $bcast(m)_u$ input to spend at least q full phases with sending status. Notice, this is exactly what $LBAlg$ requires, as by definition it has v spend the next $T_{ack} = q$ full phases after a $bcast(m)_v$ input in sending status. The t_{ack} in the lemmas statement is defined to be long enough for v to wait up to a full phase length until the next phase boundary, plus the rounds required for an additional q phases. □

We now pull together the pieces to prove Theorem 4.1:

Proof (of Theorem 4.1). The definition of the local broadcast problem had four conditions, two deterministic and two probabilistic. We consider each in turn and argue that $LBAlg$ satisfies the conditions for the parameter values specified in the theorem statement.

We first note that *timely acknowledgment* holds because $LBAlg$, by definition, has each node generate an ack in response to a $bcast$ within a fixed number of rounds that is strictly less than the t_{ack} factor from the theorem statement. Similarly, the *validity* condition holds as $LBAlg$, by definition, only has nodes broadcast messages they received in a $bcast$ input, and nodes only $recv$ messages that they actually received from another node. Moving on to the probabilistic properties, Lemma C.2 tells us that $t_{prog} = T_s + T_{prog}$ rounds, and that $t_{ack} = (T_{ack}+1)t_{prog}$ rounds. Notice, T_{ack} shows up in t_{ack} unchanged from its definition. The definition of t_{prog}, however, shows up in a form that is simplified as compared to the definition provided for T_s and T_{prog} (which contain both ϵ_1 and ϵ_2 facotrs). To match the bounds in the Theorem statement, therefore, it is sufficient to show that $T_s + T_{prog} = O\big(\log\Delta\log^2{(\frac{1}{\epsilon_2})} + r^2\log\Delta\log{(\frac{1}{\epsilon_2})}\log{(\frac{1}{\epsilon_1})}\big)$ can be asymptotically upper bounded by $O\left(r^2\log\Delta\log{(\frac{r^4\log^4\Delta}{\epsilon_1})}\right)$. We dedicate the remainder of this proof to this effort.

By definition, $\epsilon_2 \leq \epsilon_1$. We can, therefore, substitute the former for the latter in our sum, yielding:

$$T_s + T_{prog} = O\big(r^2\log\Delta\log^2{(\frac{1}{\epsilon_2})}\big).$$

We need a bound, however, that is expressed with respect the problem parameter ϵ_1. This requires us to dive deeper into the relationship between ϵ_2 and ϵ_1. There are two cases to consider given our definition above that $\epsilon_2 = \min\{\epsilon', \epsilon_1\}$. The first case is that $\epsilon_2 = \epsilon_1$. If this is true, we can simply replace ϵ_2 with ϵ_1 in our above equation, and the result is clearly upper bounded by $O\left(r^2\log\Delta\log{(\frac{r^4\log^4\Delta}{\epsilon_1})}\right)$ (which strictly increases the $\log$ factor by adding the $r^4\log^{\Delta}$ term).

The second case is that $\epsilon_2 = \epsilon' < \epsilon_1$. By definition of ϵ', it would then follow that $\epsilon_2 = \Theta\big((\frac{\epsilon_1}{r^4\log^4\Delta})^{(\gamma/r^2)}\big)$, for some constant $\gamma > 1$. The properties of γ allows us to simplify (asymptotically) the $\log^2{(\frac{1}{\epsilon_2})}$ term in our above equation as follows:

$$\log^2\big(\frac{1}{\epsilon_2}\big) = \big[\log\big(\frac{(r^4 \log^4 \Delta)^{(\gamma/r^2)}}{(\epsilon_1)^{(\gamma/r^2)}}\big)\big]^2 = \big[(\gamma/r^2)\log\big(\frac{(r^4 \log^4 \Delta)}{(\epsilon_1)}\big)\big]^2 = O\big(\log^2\big(\frac{r^4 \log^4 \Delta}{\epsilon_1}\big)\big).$$

Notice in the above we simply drop the $(1/r^2)$, as it too is bounded to be at least 1, so dropping it simply increases the value of the upper bound. Now to conclude our argument, we simply substitute this upper bound for $\log^2(\frac{1}{\epsilon_2})$ in our above sum to get the desired equation. □