MIT-CSAIL-TR-2010-029
CBCL-288

June 8, 2010

Broadcasting in Unreliable Radio Networks

Fabian Kuhn, Nancy Lynch, Calvin Newport, Rotem Oshman, and Andrea Richa

Broadcasting in Unreliable Radio Networks

Fabian Kuhn fabian.kuhn@usi.ch
Nancy Lynch lynch@csail.mit.edu
Calvin Newport cnewport@csail.mit.edu
Rotem Oshman rotem@csail.mit.edu
Andrea Richa aricha@asu.edu

Faculty of Informatics, University of Lugano, 6904 Lugano, Switzerland
Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA 02139, USA
Computer Science and Engineering, SCIDSE, Arizona State University, Tempe, AZ 85287

June 4, 2010

Abstract

Practitioners agree that unreliable links, which fluctuate between working and not working, are an important characteristic of wireless networks. In contrast, most theoretical models of radio networks fix a static set of links and assume that these links work reliably throughout an execution. This gap between theory and practice motivates us to investigate how unreliable links affect theoretical bounds on broadcast in radio networks.

To that end we consider a model that includes two types of links: *reliable* links, which always deliver messages, and *unreliable* links, which sometimes deliver messages and sometimes do not. It is assumed that the graph induced by the reliable links is connected, and unreliable links are controlled by a worst-case adversary. In the new model we show an lower bound on deterministic broadcast in undirected graphs, even when all processes are initially awake and have collision detection, and an lower bound on randomized broadcast in undirected networks of constant diameter. This clearly separates the new model from the classical, reliable model. On the positive side, we give two algorithms that tolerate the inherent unreliability: an -time deterministic algorithm and a randomized algorithm which terminates in rounds with high probability.

1 Introduction

A fundamental feature of radio networks is the presence of *unreliable* links, which sometimes deliver packets and sometimes do not. Unreliable links can be caused by radio communication *gray zones* [24], multipath propagation, and interference from unrelated networks or electromagnetic devices. As the authors note in [26], something as simple as opening a door can change the connection topology of a network, and it is common in real network deployments to occasionally receive packets from distances significantly longer than the longest reliable link [4]. Unreliable links are so pervasive that virtually every ad hoc radio network deployment of the last five years uses link quality assessment algorithms, such as ETX [13], to cull unreliable connections from those considered by higher-layer protocols. By contrast, many theoretical models of radio networks assume a fixed communication topology consisting only of reliable links.

In this paper, we explore the impact, in terms of algorithmic time complexity, of introducing unreliability into a theoretical model for radio networks. We consider a *dual graph* model, in which there are two types of communication links: *reliable* links that always deliver messages, and *unreliable* links that sometimes deliver messages and sometimes do not. The unreliable links are an abstraction that captures a variety of realistic phenomena. Our goal is to produce a model that is simple enough to be amenable to theoretical analysis, yet still captures the diversity of complex radio behaviors necessary to keep results applicable to real world deployment.

As a first step towards understanding the effects of unreliability we study the fundamental problem of network-wide message broadcast in the dual graph model. Broadcast is a powerful primitive: it can be used to simulate a single-hop network on top of a multi-hop network, greatly simplifying the design and analysis of higher-level algorithms. The broadcast problem has been extensively studied in a variety of models and settings, but mostly in reliable models (see Section 2.2 for an overview of existing work). We show that broadcast in the presence of unreliable links is strictly harder than broadcast in the reliable model. For example, in undirected reliable graphs it is possible to broadcast in rounds [2, 5], while we show that unreliable links increase the round complexity to under the same assumptions. For randomized algorithms the stretch is even worse: in the reliable model it is possible to complete a broadcast in rounds with high probability in graphs of diameter [20], while we show that there is a dual graph network of diameter 2 in which randomized broadcast requires rounds (this result appeared originally in [22] as a brief announcement). On the other hand, we show that broadcast can still be solved with reasonable efficiency in the dual graph model: we give an deterministic algorithm for broadcast in directed dual graphs, and a randomized algorithm that broadcasts in rounds with high probability. A lower bound from [11] implies that our deterministic algorithm is optimal up to a polylogarithmic factor for *directed* dual graphs; a gap remains for undirected graphs.

2 Models for Radio Networks

Many different models for wireless networks have been considered in the literature; we refer the reader to [28, 29] for a comprehensive survey. In this section we introduce our *dual graph model*. Then we briefly review several other models and explain how they compare to the dual graph model.

2.1 The Dual Graph Model

Fix any . We define a *dual graph network*, or simply a *network*, to be a pair consisting of two directed graphs, and , where is a set of nodes and . The set represents the set of reliable communication links and the set of all links, both reliable and unreliable. We assume that includes a distinguished *source node* , and that every other node is reachable in from . We call a network *undirected* if for every edge in (resp.), the edge is also in (resp.

).

We define an algorithm to be a collection of processes, which are either deterministic or probabilistic automata. (See [27] for one possible definition of automata that satisfy our requirements.) We assume that each process has a unique identifier from a totally ordered set , . We often write "process " to mean the process with identifier .

In order to define how algorithm executes on network , we must associate processes with graph nodes. Formally, our definition of an execution presupposes a bijection from to . We assume that an adversary controls the definition of . The distinction between graph nodes and processes is important for our lower bound results in Sections 4 and 6. However, we generally blur this distinction in our upper bounds in Sections 5 and 7, writing, for example, "node sends" when we really mean "process sends".

An execution of algorithm on network with a mapping proceeds in synchronous rounds, . In each round, some input may arrive at each process from the external environment. Then may or may not send a message. If it sends, its message *reaches* the processes at all of 's outgoing neighbors in , some arbitrary subset of 's outgoing neighbors in that are not outgoing neighbors in , and itself. The subset of -neighbors that the messages reaches is chosen by the adversary.

When no messages reach a process , it receives , indicating silence. When exactly one message reaches , it receives the message. When two or more messages reach , it experiences a *collision*. Collisions can be handled in several ways; we list the possible collision rules in order of decreasing strength (from the algorithmic point of view).

(CR1) If two or more messages reach (including its own message, if it sends), then receives , indicating collision notification.

(CR2) When sends, it always receives its own message, regardless of whether or not another message reaches it. (This amounts to assuming that a process cannot sense the medium while it is sending.) If two or more messages reach and does not send, then it receives collision notification ( ).

(CR3) When sends, it always receives its own message; when two or more messages reach and does not send, it hears silence ( ).

(CR4) When sends, it always receives its own message; when two or more messages reach and does not send, it receives either or one of the messages. (Which of these it receives is controlled by the adversary.)

After process receives, it changes state before beginning the next round.

Another important modelling decision is whether or not all processes start in the same round. Here we consider two rules: the *synchronous start rule* has every process begin in the first round of the execution; the *asynchronous start rule* activates each process the first time it receives a message, either from the environment or from another process.

In our upper bound results, we use the weakest assumptions, that is, collision rule CR4 and asynchronous start; our lower bounds use the strongest assumptions, collision rule CR1 and synchronous start. In each case, this serves to strengthen the results.

The definitions of executions and related concepts still make sense if algorithm is probabilistic. But now, in addition, we can define *probability distributions on executions* based on the random choices made by the processes of . To do this, we specify a particular class of (deterministic or probabilistic) adversaries. Recall that, in general, an adversary may choose the mapping, the processes that are reached by each message, and (for collision rule CR4), the particular collision behavior. An adversary class defines precisely what the adversary is allowed to choose and what information is available to it in making its choices. For algorithm and any particular adversary in the specified class, we can generate an execution probabilistically using the random choices of the processes of together with the adversary's choices. In this way, we obtain a probability distribution on executions. Then for algorithm and an entire class of adversaries, we obtain a collection of such probability distributions, one for each adversary in the class. In our lower bound results, we consider very restricted adversaries, whereas our algorithms work with respect to more powerful adversaries.

2.2 Other Models

The standard static model. The most common theoretical model for radio networks features a single network graph , which is static and captures both transmission and interference. A collision occurs at a node when two or more of its neighbors send simultaneously; typically Collision Rule 3 is assumed, that is, no collision detection is available. The communication graph may be directed or undirected.

For directed graphs with no collision detection and asynchronous start, the best deterministic upper bound known is , obtained by combining the algorithms from [20, 12], and the best lower bound is [20]. In [12] the authors give an optimal randomized algorithm that requires rounds with high probability, matching the randomized lower bounds of [23, 1], which also hold for undirected networks with synchronous start. In undirected communication graphs with synchronous start it is possible to broadcast in rounds [2, 5]. This is clearly optimal in , and [21] shows that this bound is tight even for networks of constant diameter. Interestingly, the lower bound in Section 6 applies even for undirected graphs with synchronous start, giving a clear separation between the models. The construction may appear superficially similar to the lower bound of [3], but it differs significantly (the lower bound of [3] does not apply when spontaneous wakeup is allowed).

Explicit-interference models. Several works (e.g., [15, 16]) model a network using two graphs, a *transmission graph* and an *interference graph* . It is typically assumed that . Unlike transmission edges, interference edges can only cause collisions, and messages cannot be conveyed along them. (In contrast, in the dual graph model all edges can convey messages.) A collision occurs at node when at least one of its -neighbors and at least one of its -neighbors broadcast together. The transmission and interference graphs are both static. A completely different approach is the SINR model [18, 25, 17], in which processes receive messages only when the ratio of the signal to the sum of the noise and other signals exceeds some threshold. The SINR model is geometric: the strength of the signal is assumed to degrade as a function of the distance between the processes. We refer to [30] for a recent treatment of interference in wireless networks.

Models that feature uncertainty. The closest model to the dual graph model in the literature is the dynamic-fault model of [11], in which edges of the directed communication graph can fail and recover dynamically during the execution. If one takes to be the entire graph and to be the subgraph induced by edges that never fail, the model of [11] is equivalent to dual graphs, except for one aspect: in [11] it is not assumed that is connected, and instead the broadcast is only required to reach those processes that are reachable from the source in . It is shown in [11] that deterministic oblivious algorithms require rounds to broadcast in dynamic-fault graphs; however, the notion of obliviousness used there is a very strong one, and does not allow the behavior of processes to depend on the round in which they first hear the message. In contrast, in Section 5 we give an broadcast algorithm in which processes use no information *except* the current round and the round in which they first receive the message (and their label).

	Classical model ( )		Dual graphs ( )	
SS + U	[5]	[21]	/ —	
SS + D	[20, 12]	[20]		[11]
AS + U				
AS + D				

Table 1: Bounds on deterministic broadcast

Classical model ( )		Dual graphs ( )	
[12]	[23, 1]		?

Table 2: Bounds on randomized broadcast (for any combination of assumptions with no collision detection)

The authors of [11] give a deterministic oblivious algorithm that requires rounds in dynamic-fault graphs of in-degree . This algorithm outperforms ours when ; however, it requires that all processes know (an upper bound on) the in-degree of the interference graph , whereas our algorithm requires no such knowledge.

In addition, [11] shows an lower bound for non-oblivious deterministic broadcast in directed dynamic-fault graphs. This lower bound carries over to the dual graph model, and implies that the algorithm we give in Section 5 is within of optimal for directed graphs. The authors later return to worst-case dynamic-fault graphs in [10], where they strengthen the requirement on broadcast and require it to reach all processes, even those that are not connected to the source by a fault-free path. For the stronger broadcast to be possible, it is assumed that in every round there is some functioning link between a process that has the message and a process that does not. This model does not admit a deterministic algorithm, but the authors give an expected-time randomized algorithm.

Tables 1, 2 summarize the best known upper and lower bounds for broadcast in the classical and dual graph models, assuming synchronous start (SS), asynchronous start (AS), directed (D) or undirected (U) communication graphs. Results shown in bold are presented in the current paper.

A comparison of the models. It is easy to see that the dual graph model generalizes the standard model, but what is its relation to the explicit interference model ? The explicit interference model is static, but on the other hand, the dual graph model does not feature edges that only cause interference and cannot be used to send messages. Nevertheless, the dual graph model is at least as general as the explicit-interference model, as the following easy lemma shows.

Lemma 1. *Any algorithm that broadcasts in rounds in all dual graphs of size under some collision rule CR1–CR4, also completes broadcast in rounds in all explicit-interference graphs of size under the corresponding collision rule.*

Finally, the dynamic-fault model of [11] is slightly more general than dual graphs, since it allows for the possibility of nodes that are not reachable from the source on a fault-free path.

3 The Broadcast Problem

The broadcast problem requires the dissemination of a message from the process at the distinguished source node to all processes. We assume that the message arrives at the source process prior to the first round of execution. We assume that the processes treat the message like a *black box*; i.e., that they behave the same regardless of the message contents.

We say that algorithm *solves the broadcast problem* in network provided that, in any execution of in , with any assignment of processes to nodes, the message eventually arrives at all processes. We say that *solves the broadcast problem within rounds* in network provided that, in any execution of in , with any assignment of processes to nodes, the message arrives at all processes within rounds.

Now consider a probabilistic algorithm and a fixed adversary class. Recall that generates a collection of probability distributions on executions, one for each adversary in the specified class. For any , , we say that probabilistic algorithm *solves the broadcast problem* in network *with probability* provided that the following holds: When executes in , using any adversary in the specified class, with probability at least , the message eventually arrives at all processes. We say that *solves the broadcast problem within rounds* in *with probability* provided that: When executes in , using any adversary in the specified class, with probability at least , the message arrives at all processes within rounds.

We say that network is *-broadcastable*, where is a positive integer, if there exist a deterministic algorithm and a mapping such that, in any execution of in with , with collision rule CR1 and synchronous starts, the message arrives at all processes within rounds. In other words, -broadcastable captures the intuitive notion that there is a way to resolve the contention in the network such that the message can be propagated to all nodes in rounds. Note that, if is a directed or undirected -broadcastable network, then the distance from the source to each other node in must be at most . Also, every directed or undirected network in which all nodes are reachable from the source (as we have assumed) is -broadcastable.

4 Bounds for -Broadcastable Networks

In [22], three of the authors proved the following theorem, which provides a lower bound on the number of rounds required for broadcast in an undirected -broadcastable network. In this theorem and elsewhere in this section, we assume collision rule CR1 and synchronous starts.

Theorem 2. *Let . There exists a -broadcastable undirected network such that there is no deterministic algorithm that solves the broadcast problem within rounds in .*

Proof. Let consist of an -node clique containing the source node and a "bridge" node , plus one additional "receiver" node that is connected only to . Thus, the bridge node connects the clique to . More specifically, , where , , , , , and . Let be the complete graph over . It is easy to see that the network is -broadcastable: sending followed by sending will always deliver the message to all processes.

In the executions we will consider, we assume that, in every round, the adversary resolves the communication nondeterminism as follows:

1. If more than one process sends, then all messages reach all processes and thus all processes receive .

2. If a single process at a node in sends, then its message reaches exactly the processes at nodes in . Thus, all processes at nodes in receive the message and the process at receives .

3. If only or only sends, then the message reaches all processes, so all processes receive the message.

Now assume for the sake of contradiction that there is a deterministic algorithm that solves the broadcast problem within rounds in network . Suppose that the set of process identifiers is . For every , we fix an execution of , with the communication rules listed above, in which , , and . The values of for other nodes are also fixed, according to some default rule.

Claim 3. *For every , , there is a subset such that all of the following hold:*

1. .

2. *For every , process does not send alone in the first rounds of .*

3. *For every and every , process is in the same state after rounds of and .*

Proof. By induction on . The base case, , is trivial, taking .

Assume the claim holds for , ; we show it for . By the inductive hypothesis, part (c), each process , is in the same state after rounds in all executions , . Therefore, the same set of processes send in round of all such executions. Let be the set of identifiers of these processes. If for some particular , that is, if exactly one process, process , sends, then we define . Otherwise, we define .

By construction and the inductive hypothesis parts (a) and (b), parts (a) and (b) hold for ; it remains to show part (c). Fix any process . We show that receives the same thing in round of every execution , , and therefore, using the inductive hypothesis part (c), is in the same state after round in all of these executions. There are several cases:

1. .
   Then receives in each execution , .

2. .
   Then receives in each , , because we are using collision rule CR1.

3. .
   Let . We consider subcases:

   (a) .
   Then and , so is defined to explicitly exclude . So this case cannot occur.

   (b) , that is, is the process assigned to the source node .
   Then if , then receives process 's message in each , , whereas if , then receives in each such execution.

   (c) and .
   Then is not the identifier assigned to the bridge in any of the executions , , and is not assigned to the receiver node. So if , then receives process 's message in each , , whereas if , then receives in each such execution.

   (d) , that is, is the process assigned to the receiver node .
   Then receives process 's message in each , .

Thus, in every case, receives the same thig in round of every execution , , which implies part (c). □

To conclude the proof of the theorem, we use Claim 3 for . Consider some . By Claim 3, part (b), the bridge process does not broadcast alone in the first rounds of , preventing process from receiving the message during these rounds. This contradicts the assumed time bound for . □

As shown in [22], the deterministic lower bound above can be generalized to a *probabilistic* lower bound, as follows. As before, we assume CR1 and synchronous starts.

For this theorem, we consider a restricted class of adversaries: Each adversary selects only the mapping. It resolves communication nondeterminism using the deterministic rules specified in the proof of Theorem 2. It resolves collisions using CR1.

Theorem 4. *Let . There exists a -broadcastable undirected network such that there do not exist a probabilistic algorithm and integer , , where solves broadcast within rounds in with probability greater than .*

Proof. Fix as in Theorem 2. Fix some probabilistic algorithm and integer , . Assume for contradiction that solves broadcast within rounds in with probability greater than .

For any deterministic algorithm, the proof of Theorem 2 exhibits a subset with such that, for every , process does not send alone in the first rounds of . For the probabilistic algorithm , each way of fixing the random choices (using a predetermined choice sequence) yields a deterministic algorithm, and so also yields a subset with the same properties with respect to the defined for these fixed choices.

From the probability distribution of random choices, we derive a probability distribution of subsets , each with at least elements.

Claim 5. *There is some such that, with probability at least , the subset derived from the probability distribution of random choices includes .*

Proof. Let be the set of -element subsets of . For any and , define if , otherwise. Then for any , . For any , write for the probability that . Then

Now suppose for contradiction that for every , the probability that the derived contains is strictly less than . That means that, for every ,

Then

Thus,

which is a contradiction. □

Now fix as in Claim 5 ; that is, such that, with probability at least , the derived subset includes . Then process does not broadcast alone in the first rounds of any of the executions associated with the random choice sequences that give rise to subsets that include . Note that all of these are executions in which , , and , and the other values of are determined by a default rule.

Now define the adversary to fix , , and , and to determine the other values of by the same default rule. Then when executes with this adversary, with probability at least the random choices prevent process from broadcasting alone in the first rounds. Because is associated with the bridge node, it follows that with probability at least , the message does not get to process within rounds. This contradicts the success probability assumption for . □

Notes: The deterministic lower bound in Theorem 2 is matched by a deterministic round-robin broadcast strategy, which succeeds in rounds in (directed or undirected) graphs of constant diameter, and hence, in -broadcastable networks for any constant .

Bar-Yehuda et al. [2] proved a linear-round lower bound for deterministic broadcast in certain -broadcastable networks. Their proof uses our collision rule CR4, which allows nondeterministic collision resolution without collision detection. This gives the adversary more power than CR1 as used in our proof of Theorem 2. Kowalski and Pelc [] observed that the lower bound of [2] does not work with CR3, which provides deterministic collision resolution without collision detection; in fact, they presented an -round deterministic solution, for the particular graphs used in [2]. They also presented probabilistic algorithms for these particular graphs, using CR4.

5 Deterministic Upper Bound

We describe a deterministic algorithm that solves the broadcast problem in time. To strengthen the upper bound we assume the weakest assumptions from Section 2: a directed dual graph, Collision Rule 4, and asynchronous start. For simplicity we assume that , is a power of 2, and that , where is the unique id set in our model.

Our algorithm follows the standard broadcast strategy of cycling through *selection objects* of exponentially increasing sizes; c.f., [6, 7]. A selection object is a broadcast schedule for every node, parameterized by the number of nodes participating, which guarantees that if the correct number of nodes participate, each node will be isolated and will be the only node to broadcast in some round. Broadcast algorithms that follow this strategy are typically concerned with isolating all *frontier nodes*, nodes adjacent to some node that does not have the message yet.

In the reliable model, when a frontier node is isolated and broadcasts alone, all of 's neighbors receive the message. Thereafter, node is no longer a frontier node; even if continues broadcasting, its transmissions cannot interfere with the progress of the message, because all its neighbors already have the message. Thus, in the algorithms of, e.g., [6, 7], nodes continue to cycle through selective families forever, and never stop broadcasting. The different selector sizes are used to ensure that at least one selector matches the size of the frontier, ensuring that all frontier nodes will be isolated.

In the dual graph model the situation is more complicated; there is no clear-cut "frontier". Suppose that node has some -neighbors that have not received the message, but all of its -neighbors already have the message. Informally, node no longer contributes to the progress of the algorithm, because the adversary can prevent it from getting the message out to new nodes (its -neighbors); in this sense is no longer a frontier node. However, can still *interfere* with the progress of the algorithm, because its broadcasts can cause collisions at nodes that do not have the message. Due to this difficulty, we allow processes to participate in each selection object exactly once, limiting the interval during which they can cause interference. This strategy has the additional advantage that nodes eventually stop broadcasting. It requires, however, a more nuanced argument to establish the message's progress.

In the following, we use the notation , where , to indicate the interval , and use , where , to indicate . We continue by defining *Strongly Selective Families* (SSFs), the selection objects used in our algorithm.

Definition 6 ([8]:)**.** *Let . A family of subsets of is -strongly selective if for every non-empty subset of such that and for every element there is a set in such that .*

Erdös et. al. provide an upper bound on the size of these objects [14]:

Theorem 7 ([14])**.** *For any and for , there exist -strongly selective families of size .*

Let . For each , let be an -SSF of size , where . (By [14] we know such families exist.) We fix some total ordering on the sets that comprise each family . Furthermore, we assume that is the round robin sequence, which isolates every node in the graph. Thus, is an -SSF. (We can assume this because .) We now define our algorithm, which we call *strong select*.

The Strong Select Algorithm Assume without loss of generality that nodes have a access to a global round counter.[1] The algorithm divides the rounds into contiguous groups of length called *epochs*. The first round of each epoch is dedicated to the smallest SSF ; the next two rounds are dedicated to ; the next four rounds to , and so on. In general, we go through sets of each SSF in each epoch.
When a node first receives a message, it waits, for each , until cycles back to . It then *participates* in the SSF for a single iteration, broadcasting in any round in which its id is included in the corresponding SSF set. That is, after it starts participating, in round of epoch , a node with id broadcasts iff . After participating in one complete iteration of an SSF, the node stops participating in that family. Each node participates in exactly one iteration of each SSF used in the algorithm.

For a given SSF , we use the term *iteration* to describe a complete cycle through . Note that each iteration of is spread out over epochs. We also remark that in a given epoch it could happen that a node participates in some selector families but not in others, because it is waiting for those other selector families to cycle back to their first set.

Analysis. Fix a network and an execution of the algorithm in the network. Define to be the log-factor in the size of the SSFs: formally, is a function such that and for each SSF used by the algorithm, .

The proof involves an amortization argument, where (roughly speaking) we show that in every sufficiently long interval the algorithm always makes progress: either many new nodes receive the message for

[1]To see why this is without loss of generality, note that the source can label every message it sends with its local round counter. When any other node is first activated by receiving a message, it adopts the round number on the message, and labels all future broadcast messages with its local round counter.

the first time, and a lot of progress is made; or few nodes receive the message for the first time, but then these nodes only have to contend with each other, and they will quickly be isolated and get the message out to other nodes. To formalize this, we define the *density* of an interval , denoted , to be the number of nodes that receive the message for the first time in the interval, divided by :

# nodes that receive the message
for the first time during (1)

Given an SSF , let denote the number of complete iterations of that fit in the interval . Finally, we fix two constants that are used throughout the proof: we define a density threshold

and let be the smallest round such that , that is, the round in which the density over the entire execution first drops below . We will eventually show that the algorithm terminates no later than round .

We begin by showing that each node that participates in one of the last iterations of some SSF ending by round is isolated.

Lemma 8. *Consider the last iterations of in the interval , for some . Every nodes that participates in one of these iterations broadcasts alone at some point during the iteration.*

Proof. Let be the number of nodes that participate in one of the last SSFs. Let be the number of rounds required to complete an iteration of : family contains sets spread out over epochs (with sets from in each epoch), and each epoch requires rounds. Any node that participates in one of these iterations must receive the message for the first time in the interval where . Therefore, if we denote by the number of nodes that receive the message for the first time in , then . Note also that , otherwise we would have and would not be minimal. It follows that

(1)

We have shown that the *total* number of participants in any of the last iterations is less than ; therefore, the number of participants in each individual iteration is also less than (because each node participates in just one iteration). From the definition of an SSF, each participant in any of the last iterations will be selected to broadcast alone in the network. □

Lemma 9. *No node receives the message for the first time in the interval , where .*

Proof. If one or more nodes receives a message in this interval, then , contradicting the minimality of . □

Theorem 10. *The strong select algorithm solves broadcast in rounds in any directed (or undirected) network , with collision rule and asynchronous starts.*

Proof. We first show that every node receives the message by the end of round .

Assume for contradiction that some node has not received the message by round . Since all nodes are reachable from the source in , there exist two nodes such that has the message by round and does not, and . This means that node has not been isolated prior to round ; we will show that node cannot have received the message prior to round , deriving a contradiction. The proof involves repeatedly using Lemma 8 to show that node cannot have received the message by the last iteration of selector families of decreasing size, pushing forward the round in which node first received the message until eventually we exceed round , obtaining a contradiction to Lemma 9.

Formally, we show by backwards induction on that for all , node did not receive the message by round . Here, as in the proof of Lemma 8, we define to be the number of rounds required for a complete iteration of . Note that may be negative, in which case the claim trivially holds.

Induction base: for , suppose that and suppose by way of contradiction that node received the message by round . Since cycles back every rounds, node started participating in no later than round ; by round it has had enough time to participate in a full iteration of . However, recall that is an -SSF; any node that participates in a full iteration of is isolated. Since we assumed that has not been isolated by round , it cannot have received the message by round .

Induction step: suppose that node did not receive the message by round , and suppose by way of contradiction that received the message by round . Observe that since and , we have : two iterations of fit inside every iteration of . Since node did not get the message by round , and we assumed for contradiction that it got it by round , it participates in one of the last iterations of . From Lemma 8, node is isolated, yielding a contradiction. This concludes the induction.

We have shown that node did not get the message by round . Since we assumed that did get the message prior to round , it follows that got the message for the first time in the interval , contradicting Lemma 9. This completes the first part of the proof; we can now conclude that every node receives the message no later than round .

To conclude the proof, consider the interval , where we define . If , then nodes receive the message during the interval . On the other hand, if , then by definition , so again all nodes receive the message no later than round . In both cases the broadcast is complete by round , and the algorithm terminates in rounds. □

A Note on Constructive Solutions The -SSFs of size used in *strong select* are derived from an existential argument [14]. The smallest-size constructive definition of an -SSF, from a 1964 paper by Kautz and Singelton [19], is of size . Replacing the SSFs in our algorithm with the variant from [19] would increase our time complexity by only a -factor.

6 Deterministic Lower Bounds

In this section, we present two lower bounds for deterministic broadcast algorithms. For both algorithms, we assume collision rule CR1 and synchronous starts. The following bound is a straightforward adaptation of the result presented as Theorem 4.2 of [9]:

Theorem 11. *There exists a* ‾*-broadcastable directed network* , *such that every deterministic algorithm that solves the broadcast problem in has an execution in which it takes rounds until the message arrives at all processes.*

It follows that our upper bound in Section 5 is tight to within a factor of ——. However, this lower bound construction depends heavily on the fact that the network is directed. If the graph were undirected, processes could provide feedback to their neighbors when they receive the message; this would break the reduction to the SSF lower bound which is at the core of the lower bound from [9].

We proceed with an lower bound that handles *undirected* networks. It remains an open question whether this bound is tight.

Theorem 12. *There exists an undirected network* , *such that every deterministic algorithm that solves the broadcast problem in has an execution in which it takes rounds until the message arrives at all processes.*

In the following proof, we say a process is *about to be isolated* after a given finite execution if it will send in the next round, and is the only process that will do so.

Proof. Let the set of nodes be , where is the source node. We assume for simplicity that is a power of , . We divide the nodes into *layers* , —, where and for each , —, .

We construct a dual graph with vertex set . The reliable graph, , is a complete layered graph, with edge set given by:

and

and

The unreliable graph, , is the complete graph over : . Note that by design, when process trasmits, where , its message can reach the processes at any subset of the nodes that includes (if) and (if —).

We assume that the identifier set includes a distinguished identifier that is assigned to node , that is, that .

We construct an execution and mapping in stages numbered —. At Stage , —, the construction assigns processes to the nodes (and) in layer , and constructs a longer prefix of . For any , let be the set of identifiers of processes that are assigned to nodes in layers , by the end of Stage . Our construction will ensure that, by the end of , exactly the processes with identifiers in have received the broadcast message. Moreover, ends with some process in about to be isolated.

As a base case for this construction, in Stage we construct an execution in which all -edges are used in every round, ending with the first round after which is about to be isolated. There must be some such round, since otherwise no process other than process will ever receive the message. We define . Note that by the end of , only has the message, because it has not yet sent alone.

Now we describe Stage , —, which assigns processes to the two nodes (and) in layer , and extends to . For each pair of processes , we define an extension of , in which we assign processes and to , arbitrarily assigning one of the two processes to and the other to . We first define for any , and then describe how we choose the particular pair that is used to construct . For convenience we number the rounds of after as .

In round of , we know that exactly one process sends, and it belongs to . The adversary allows this message to reach (and so, to be received by), exactly the processes in (by using the appropriate edges). Thereafter, we use the following adversary rules to determine where messages reach. Collisions are handled according to CR1, our strongest rule.

1. If more than one process sends, then all messages sent reach everywhere, and all processes receive .
2. If a single process sends alone, then its message reaches exactly the processes with ids in , so exactly these receive it.
3. If a single process sends alone, then the message reaches all processes, so they all receive it.
4. If either or sends alone, then the message reaches all processes, so they all receive it. (We include this rule for completeness; this case will not arise within the number of rounds we will consider.)
5. If no process sends, then all processes receive .

These rules are designed so that, until either or sends alone, only the nodes in will have the broadcast message. It is easy to verify that the adversary can *always* follow the rules above regardless of the process assignment to nodes (which we have not yet committed to at this point).

Having defined for all possible pairs , we must choose the pair that will actually be assigned to layer and used to define . We do this by constructing a sequence of *candidate sets* of process identifiers, , where , and each candidate set in the sequence is a subset of the previous one. Informally speaking, the identifiers in each are the candidates that remain after we take into account behavior through round . The process ids and will be elements of .

We begin with and construct the remaining candidate sets inductively. Observe that —, because we apply this construction for only — stages and add only two processes to at each stage.

We maintain the following inductive property for each candidate set (where).

Property

(1) —.

(2) Let , and let and be two pairs of elements of . Suppose that is either in neither subset or in both. Then process receives the same values (either , , or an actual message) in rounds of and .

(3) Let . Then neither nor sends alone at any of rounds of .

Part (1) of will be used to ensure that we can extend Stage to rounds. Part (2) ensures that neither of the processes assigned to layer learns the identity of the other process, and also that none of the processes assigned to layers greater than learns the identities of the processes assigned to layer . Part (3) says that the candidates that remain after round have not yet sent alone, after .

Suppose we already have a set satisfying . Conditions (1) and (3) together imply that there exist such that neither nor sends alone in any of rounds of . We arbitrarily choose one such pair , and define to be the prefix of ending at the first time when either or is about to be isolated; this extends by at least rounds.

Inductive construction of $n-$ $-$ Property is clearly true for . Suppose we have already constructed , where , such that holds, and let us construct . We begin by defining two sets:

is the set of remaining candidates such that if we assign to layer , then will send in round . Formally, is defined to be the set of ids such that for some , , process sends in round of . (By Part of , this set is equivalent to what what we obtain if we replace "for some " with "for every ".)

is the set of remaining candidates that will send in round if we do *not* assign them to layer . That is, is the set of nodes such that for some where , process sends in round of . (As above, by Part of , this also holds if we replace "for some " with "for every ".)

Note that for every , process will not send in round *regardless* of whether or not it is assigned to layer .

Now we are ready to define . We consider cases based on the sizes of and .

Case I: , that is, there are at least two processes that would send in round if they are not assigned to layer .

In this case we omit two such processes from the candidate set: we define , where are the two smallest elements of .

Case II: and —. Then we set .

Case III: and —. Then we set .

That is, if at least two processes would send in round if they *did not* receive the message in round , then we omit two such processes from the new candidate set. This guarantees that, in the remaining executions we will consider, they will not receive the message in round and will therefore send in round , so everyone will receive in round .

On the other hand, if at most one process would send in round if it did not receive the message in round , then we determine the candidates based on the number of processes that would send in round if they *did* receive the message in round . If at least half would send in round , we include exactly those that would send. This ensures that, in the remaining executions, at least two of these will receive the message in round and will send in round , again causing everyone to receive in round .

The remaining case is where at most one process would send in round if it did not receive the message in round , and strictly fewer than half would send in round if they did receive the message in round . In this case, we include exactly those that would not send if they received the message, omitting the possible single process that would send if it did not receive the message. This ensures that, in the remaining executions, the processes that receive the message at slot will not send at slot . Other processes, however, may send at slot .

Claim 13. *Property holds for . That is,*

1. —.

2. *Let , and let and be two pairs of elements of . Suppose that is either in neither subset or in both. Then process receives the same values (either , , or an actual message) in rounds of and .*

3. *Let . Then neither nor sends alone at any of rounds of .*

Proof. For Part 1, note that —, by Part 1 of . If is even, the result then follows by easy calculations based on the three cases in the definition of from . If is odd, then the calculation is straightforward for Cases 1 and 2(a). The argument for Case 2(b) is slightly more involved. We know that —. We know that — is even, because . Since is odd, we have — . Also, since —, we have —. So we have

By the lower bound on , the right-hand side is

as needed.

Part 3 follows from Part 3 of and the cases in the definition of .

In remains to show Part 2; for this, fix as in the hypotheses. Part 2 of implies that receives the same values in the first rounds; we consider what happens in round . We consider cases as in the definition of .

Case I: . Then in both and , two processes in do not receive the message in round and so send at round . It follows that receives in round in both executions.

Case II: and —. Then both and send in round in and both and send in round in , so again receives in round in both executions.

Case III: and —. Here we must carefully consider which processes send in round . We know that neither nor sends in round of , and neither nor sends in round of . Also, we know that each process in chooses whether/what to send based on its own state after , its receipt of the message in round , and whatever values it receives in rounds . All of this information is the same in and , using Part 2 of Property (here, each element of is always in neither of the two sets). Therefore, it behaves the same in round of both executions.

We now consider two sub-cases.

Subcase IIIa: . Then no process in sends in round of , and no process in sends in round of . Since neither nor sends in round in , and neither nor sends in round in , it follows that in this subcase, no process in sends in round of or .

We are left to consider the processes in . If no process in sends in round then receives in both executions. If two or more processes in send in round , then by the adversary rules, both messages reach all processes, so receives in both executions. If exactly one process in sends, then by the adversary rules, the message reaches exactly the processes in in , and reaches exactly the processes in in . Since is either in both sets and or neither, the message reaches either in both executions or in neither execution. Thus, either receives the message in both executions, or it receives in both executions.

Subcase IIIb: **.** Then a single process sends in round of both executions. This follows because we have explicitly omitted from , ensuring that it does not receive the message in round in or , which implies that it sends in round . By the adversary rules, we know that 's message reaches all processes, hence reaches , in both executions.

Now we consider the processes in . If no process in sends in round , then receives the message from in round in both executions. If one or more processes from sends, then by the adversary rules, their messages reach all processes. So then receives in both executions (because the message(s) collide with the message).

Combined, these cases establish Part 2 of , thus completing the proof of the claim. □

Claim 13 implies that holds for . Therefore, there exist two identifiers such that neither nor sends alone at any of the first slots of . (Use Part 1 to show that , and Part 3 to show that the processes in this set do not send alone.) We then define to be the prefix of that ends just before the first round where either or sends alone. This gives us an extension of at least slots. Note that only processes in have the broadcast message by the end of .

For the entire construction, we begin with and construct successive extensions . Since only two new processes receive the message in each stage, by the end of , some processes have still not received the message. The resulting execution is rounds long, which yields our lower bound. □

7 Randomized Upper Bound

In this section we give a simple randomized algorithm for broadcast, which completes in rounds with high probability. We assume a directed communication graph, asynchronous start, and collision rule 4, the weakest rule.

Algorithm Harmonic Broadcast Nodes begin participating immediately after they receive the message. If node receives the broadcast message for the first time in round , then in all rounds it transmits the message with probability , where

where is an integer parameter that will be fixed later.

Hence, for the first rounds after receiving , nodes transmit the message with probability ; in the next rounds the message is transmitted with probability , then the probability becomes , and so on. For , we define . For convenience, we assume that the sender receives at time , i.e., and starts broadcasting in round .

In order to intuitively see why the algorithm works, consider a layered network with layers of different sizes, where all nodes in a layer receive the message at the same time and where the message is propagated to the next layer as soon as some (specific) node in the current layer broadcasts alone. All nodes in the same layer always broadcast with the same probability. If a layer contains nodes, nodes need tosend with probability roughly times to guarantee progress with high probability. Setting , the described algorithm guarantees exactly this for all values of . As small layers need to make progress faster, the algorithm starts with large probabilities and gradually decreases them. Since each probability is used times, it is clear that the algorithm requires at least rounds to complete. The second -factor in the bounds is a consequence of interference from 'old' layers. To intuitively see this, consider a layered network in which all layers have constant size. If we would get to a new layer every rounds, the sum of the broadcast probabilities of previous layers could be estimated by a harmonic sum and would thus be of order . However, this sum needs to be before the currently active layer can make progress with reasonable probability. This leads to larger intervals of roughly rounds on average between reaching successive layers.

Analysis

For every , we define

$$ \tag{2}$$

to be the sum of the transmitting probabilities in round . We say that round is *busy* if , and otherwise we say that round is *free*. We begin by bounding the number of busy rounds in any execution from above.

We define the *wake-up pattern* of an execution to be a non-decreasing sequence of round numbers, where , and is the round in which the node receives the message. (That is, is the round in which the first node that is not the source receives the message, and so on.) Note that the wake-up pattern of an execution determines the broadcasting probabilities of the nodes in every round; therefore, to reason about broadcast probabilities it is sufficient to reason about all possible wake-up patterns (including ones that cannot occur in any execution of the algorithm).

Lemma 14. *Let be the maximum number of busy rounds induced by any wake-up pattern. Then there is a wake-up pattern for which rounds are all busy.*

Proof. Let be a wake-up pattern that maximizes the number of busy rounds, and among those wake-up patterns that maximize the number of busy rounds, minimizes the number of free rounds before the last busy round. We argue that this wake-up pattern has no free rounds between the busy rounds, that is, rounds are all busy rounds.

For the sake of contradiction, suppose that there is a free round before the last busy round, and let be the last free round before the last busy round. By definition, , and since round must be busy, we also have . The sum of the broadcast probabilities can only increase from one round to the next if some new node receives the message for the first time; thus, there is some node such that .

Consider the alternative wake-up pattern , where if and otherwise . Let us use to denote the sum of the probabilities induced by wake-up patterns and in round , respectively. Further, let be the sending probability in round of a node that first receives the message in round (as defined in the algorithm). Because the wake-up patterns are the same up to round , we have for all . For , we have

Therefore, if round is busy for , then round is busy for , and the total number of busy rounds in is at least the same as in . Furthermore, round (which was free for) is busy for , because round is busy for . It follows that has fewer free rounds before the last busy round than does, but it has at least as many busy rounds, contradicting the choice of . (Recall that was chosen to be a wake-up pattern that maximizes the total number of busy slots, and among these wake-up patterns, minimizes the number of free time slots before the last busy slot.) □

The following lemma bounds the total number of busy rounds induced by any wake-up pattern.

Lemma 15. *The total number of busy rounds for any wake-up pattern is at most* .

Proof. Consider an arbitrary -node wake-up pattern . We show that there has to be a free round by time where , denotes the harmonic sum. Together with Lemma 14, this implies the claim.

We prove that there is a free time round by time by induction on . For the claim is immediate.

Thus, let . For , let be the node that wakes up (receives the message) at time , and let be the first free round when using the -node wake-up pattern (that is, the prefix of in which nodes are never awakened). By the induction hypothesis, for all . We want to show that .

Let us first consider the case where for some . In this case, round remains free when we consider the complete wake-up pattern ; thus, .

Next, consider the case where for all . For any , at time , the sending probability of node is

For the sum of transmitting probabilities, we therefore obtain

Hence, round is free, as required. □

We say that a process is *isolated* in a round if it is the only process transmitting in that round. In the following, we show that a process that broadcasts in a free round is isolated with high probability, and that as soon as the number of free rounds since a process received the message is large enough, that process is isolated with high probability.

Lemma 16. *Let be a free round and assume that node transmits in round with probability . The probability that is isolated in round is at least* .

Proof. Because is a free round, all transmitting probabilities are smaller than and thus for all we have . Let be the probability that none of the nodes in send in round . We have

In the last two steps we used the fact that for it holds that , and that , because is a free round. The probability that is isolated in round is . □

Lemma 17. *Consider a node , and let be the time when first receives the message. Further, let be such that at least half of the rounds are free. If for some , then with probability larger than there exists a round such that is isolated in round .*

Proof. Let . Note that because sends with probability in the first rounds (a nd hence the first rounds are not free). In round , the transmitting probability of is

$$ \tag{3}$$

Because the transmitting probability is non-increasing, by Lemma 16, for every free round , the probability that is isolated is larger than . Let be the probability that there is no free round in which transmits alone. As there are at least free rounds, the probability is bounded by

The first inequality follows from Lemma 16 and from (3); the second inequality follows because for all we have . Finally, the third and fourth inequalities follow from and from the fact that , respectively. □

Finally, we are ready to prove the following main theorem:

Theorem 18. *If for some , all nodes of the network receive by time with probability at least .*

Proof. For any node , let be the round in which first receives the message, or if never receives the message. Let be the first round after in which the number of free rounds greater than is equal to the number of busy rounds after . By Lemma 17, node has been isolated by round with probability at least . By a union bound argument, the probability that *every* node has been isolated by (assuming is finite) is at least . We will show that whenever this event occurs, all nodes receive the message before the first time in which the total number of free rounds in the execution equals the total number of busy rounds. Together with Lemma 15, this proves the theorem.

Let be the first round in which over the entire interval , the number of free rounds equals the number of busy rounds, and suppose by way of contradiction that every node was isolated no later than round (if round is finite) but some node has not received the message. Let be the non-empty set of nodes that have not received the message by round . Since is broadcastable, there exists a directed edge where and . If we can show that , then by our assumpion that is isolated by round , process receives the message by round , contradicting the choice of .

To that end, assume by way of contradiction that (or is infinite), that is, the number of free rounds in the interval is smaller than the number of busy rounds. By choice of we know that the number of free rounds in the interval is at least the number of busy rounds in the interval . It follows that the number of free rounds in exceeds the number of busy rounds in , contradicting the minimality of . □

By setting , we get , and hence we have shown that

Theorem 19. *The randomized broadcast algorithm solves broadcast in rounds with probability at least . in any directed (or undirected) network , with collision rule 4 and asynchronous start.*

8 Conclusion

In this paper we introduce dual graphs, a new model for radio networks. Unlike most traditional models for radio networks, the dual graph model allows for *dynamic* interference and unreliable communication. Like traditional models, the dual graph model includes a graph of reliable communication links; but in addition, unreliable links are represented in the form of a second graph , whose edges can be deployed against the algorithm by a worst-case adversary. Algorithms for the dual graph model are therefore highly resilient to interference, noise, and unpredictable communication links.

In the current paper we showed that for the broadcast problem, resilience to link failures comes at the cost of higher round complexity: a lower bound of holds for a setting in which the traditional model admits an -round deterministic algorithm. Our deterministic upper bound, at rounds, does not yet match this lower bound; nevertheless, we gave reasonably efficient deterministic and randomized algorithms for broadcast.

A significant part of the difficulty comes from the fact that the network topology is unknown to the processes at the time of the broadcast. In future work it is our intention to explore *repeated* broadcast in dual graphs, where we hope to improve long-term efficiency by learning the topology of the graph. Topology control in dual graphs is another interesting area for future research.

References

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43(2):290–298, 1991.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in radio networks: an exponential gap between determinism randomization. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 98–108, New York, NY, USA, 1987. ACM.

[3] D. Bruschi and M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distrib. Comput.*, 10(3):129–135, 1997.

[4] K.-W. Chin, J. Judge, A. Williams, and R. Kermode. Implementation Experience with MANET Routing Protocols. *SIGCOMM Computer Communication Review*, 32(5):49–59, 2002.

[5] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. In *Symposium on Discrete Algorithms*, pages 861–870, 2000.

[6] M. Chlebus, L. Gasieniec, A. Ostlin, and J. Robson. Deterministic broadcasting in radio networks. In *the International Colloquium on Automata, Languages and Programming (ICALP)*, 2000.

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

[8] A. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *the annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 709–718, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[9] A. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *Journal of Parallel Distributed Computing*, 64:89–96, 2004.

[10] A. E. F. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Broadcasting in dynamic radio networks. *J. Comput. Syst. Sci.*, 75(4):213–230, 2009.

[11] A. E. F. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *J. Parallel Distrib. Comput.*, 64(1):89–96, 2004.

[12] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. *J. Algorithms*, 60(2):115–143, 2006.

[13] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Wireless Networks*, 11(4):419–434, 2005.

[14] P. Erdos, P. Frankl, and Z. Furedi. Familes of finite sets in which no set is covered by the union of others. *Israel Journal of Mathematics*, 51:79–89, 1985.

[15] F. Galčík. Centralized communication in radio networks with strong interference. In *SIROCCO '08: Proceedings of the 15th international colloquium on Structural Information and Communication Complexity*, pages 277–290, Berlin, Heidelberg, 2008. Springer-Verlag.

[16] F. Galčík, L. Gasieniec, and A. Lingas. Efficient broadcasting in known topology radio networks with long-range interference. In *PODC '09: Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 230–239, New York, NY, USA, 2009. ACM.

[17] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer. Local broadcasting in the physical interference model. In M. Segal and A. Kesselman, editors, *DIALM-POMC*, pages 35–44. ACM, 2008.

[18] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on information theory*, 46:388–404.

[19] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

[20] D. R. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. *Distrib. Comput.*, 18(1):43–57, 2005.

[21] D. R. Kowalski and A. Pelc. Time complexity of radio broadcasting: adaptiveness vs. obliviousness and randomization vs. determinism. *Theor. Comput. Sci.*, 333(3):355–371, 2005.

[22] F. Kuhn, N. Lynch, and C. Newport. Brief announcement: Hardness of broadcasting in wireless networks with unreliable communication. In *The Annual ACM Symposiun on Principles of Distributed Computing (PODC)*, pages 330–331, 2009.

[23] E. Kushilevitz and Y. Mansour. An lower bound for broadcast in radio networks. *SIAM J. Comput.*, 27(3):702–712, 1998.

[24] H. Lundgren, E. Nordstr
"o, and C. Tschudin. Coping with Communication Gray Zones in IEEE 802.11b Based Ad Hoc Networks. In *the International Workshop on Wireless Mobile Multimedia*, 2002.

[25] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM*. IEEE, 2006.

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

[27] C. Newport and N. Lynch. Modeling Radio Networks. In *the International Conference on Concurrency Theory (CONCUR)*, 2009.

[28] D. Peleg. Time-efficient broadcasting in radio networks. In *DISC '07: Proceedings of the 21st international symposium on Distributed Computing*, pages 3–4, Berlin, Heidelberg, 2007. Springer-Verlag.

[29] S. Schmid and R. Wattenhofer. Algorithmic models for sensor networks. In *IPDPS*. IEEE, 2006.

[30] P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Algorithmic models of interference in wireless ad hoc and sensor networks. *IEEE/ACM Trans. Netw.*, 17(1):172–185, 2009.

Appendices

A Comparison of Explicit-Interference Models and Dual Graphs

In this section we prove Lemma 1. The corresponding collision rules for explicit-interference graphs are defined the same as the originals, with the following modification: all messages sent by such that reach node ; however, if , then node cannot under any circumstances receive messages sent by . If the only message that reaches node was sent by , then node receives .

Proof of Lemma 1. . We prove the claim for undirected graphs; for directed graphs the proof is similar and slightly easier.

We show that the behavior of the adversary in an explicit-interference model can be simulated by an adversary for the dual graph model. More specifically, given an explicit-interference graph of size , we show that a dual-graph adversary for the dual graph , where and , can cause all nodes to receive exactly the same feedback that they would receive in the original graph. The proof is not tied down to a specific collision rule; we map every possible behavior of the adversary to the same behavior, so the proof works for all collision rules.

In a given round, we partition the nodes based on their behavior and the feedback they receive. Let be the set of nodes that broadcast in the round. Next, let be the set of nodes that receive a message (that they did not broadcast), and let be the nodes that receive collision notification. All the remaining nodes (in) hear only silence.

Recall that we chose , so all edges are controlled by the adversary we are constructing. We schedule only edges that were involved in a collision, that is, edges such that

(1) there exists such that .

(2) , and

(3) .

In other words, we choose edges such that some message (sent by some) reaches node , but node does not receive a message; and in addition, node sends, so it can be (at least partially) blamed for the collision.

First, observe that whenever two or more messages reach node in the original graph, the same messages will reach node in the dual graph. Also, messages sent along -edges reach the same nodes in both cases. We show that it is possible for the dual-graph adversary to provide exactly the same feedback as the explicit-interference adversary to all nodes.

I. Let . Then there are two cases.

   (a) Node has exactly one -neighbor and all its -neighbors are not in . In this case the dual-graph adversary does not use any of 's -edges (since they are not involved in a collision), so 's message is the only message that reaches . The dual-graph adversary is free to deliver 's message to .

   (b) Node has at least one -neighbor and at least one -neighbor . The explicit-interference adversary must have used collision rule 4 to deliver 's message to . The dual-graph adversary is free to do the same.

II. Let . Then at least two messages reach in the original graph, and therefore also in the dual graph. The adversary can provide with collision notification.

III. Let (that is, hears silence). Then there are two cases.

   (a) At least two messages reach in the original graph. As already explained, the same messages will reach in the dual graph. Since the explicit-interference adversary did not provide with collision notification, it is permissible for the dual-graph adversary to do the same (under the corresponding collision rule).

   (b) No -neighbor of broadcasts. Our only concern in this case is that might now receive a message along the -edges that the dual-graph adversary schedules (where previously such edges could only cause interference). However, the dual-graph adversary only deploys a -edge adjacent to if some -neighbor of broadcasts. Therefore no -edges of are deployed.

IV. Let . Node receives either its own message, or collision notification (under collision rule 1). If receives its own message, then it is easy to verify that the dual-graph adversary can do the same. If receives collision notification, then at least two messages reach it in the original graph; as before, the same messages reach it now, so the dual-graph adversary is free to provide collision notification.

□