Massachusetts Institute of Technology
6.895: Advanced Distributed Algorithms
Professor Nancy Lynch

Lecture 10
March 15, 2006

Network-Wide Broadcast

These notes cover the first of two lectures given on the topic of broadcast. These notes are divided as follows. Section 1 is an introduction to broadcast, outlining problem and summarizing the results achieved in the papers. Section 2 discusses various broadcast models for deterministic and randomized broadcasting. Section 3 covers the first BGI paper, which describes a randomized broadcast protocol and argues an (incorrect) linear lower bound on deterministic broadcast. Section 4 covers the second BGI paper, which presents an efficient randomized emulation algorithm to emulate any algorithm for a single-hop network with collision detection on a multihop network without collision detection.

1 Introduction

Broadcasting is the problem of propagating a message from a known source node i_0 to all the nodes in a network. The network is modelled as a connected undirected graph, in which there is an edge between two nodes (they are neighbors) iff they can communicate directly via single-hop broadcast. Broadcast is a common task for real sensor networks (and other ad hoc networks) and has also been studied extensively by theoreticians. The papers we'll consider are mainly theory papers; in addition to trying to understand the theoretical results, we should try to understand what the results mean in practice—e.g., are the assumptions reasonable?

The broadcast problem is greatly simplified if the network layout is known, so all the papers assume unknown network topology—nodes may just know local information—typically their own ids and (usually) those of their neighbors. The papers also make strong assumptions about what happens when collisions occur:

- The receiver definitely does not receive any message
- The receiver cannot tell that anything was sent (can't distinguish collision from nothing sent)

Actually, the lower bound in the BGI-1 paper is erroneous as stated—the construction assumes that one of the possible outcomes when a collision occurs is that a receiver might receive one of the messages correctly. This error was apparently not discovered until years later, by Kowalski and Pelc. BGI note, however, that their result still holds for a slightly different model, in which this kind of collision behavior is also possible.

The first four papers attempt to get close upper and lower bounds for the amount of time that it takes to complete a broadcast. In the deterministic case, this is worst-case time. In the randomized case, they talk about expected time, or about high probability of completing by a certain time. The papers also want to show that randomized algorithms perform noticeably better than deterministic ones.

The main results of the broadcast papers are as follows:

1. (BGI-1) An upper bound on randomized broadcast.
   For any ϵ, there is a randomized protocol achieving bcast in $O((D + \log(n/\epsilon)) \log n)$ slots, with probability at least $1 - \epsilon$, where $D =$ max distance between source and any other node, $n =$ total number of nodes in the network.
2. (BGI-1) A lower bound on deterministic broadcast.
   Any deterministic bcast protocol requires $\Omega(n)$ slots, even if network diameter is constant. But (caution) this holds just with the weaker collision assumption (Model B, see next section). They claim that they have shown an exponential gap between randomized and deterministic protocol, in Model A, but this doesn't show it. Kowalski-Pelc claimed this proof recently.
3. (BGI-2) An algorithm for randomized broadcast.
   They present a randomized algorithm that allows emulation of any algorithm that is designed for a single-hop radio network with collision detection, in a multi-hop network with no CD, while keeping the costs reasonably bounded.
4. (Kowalski-Pelc) An upper bound on deterministic broadcast.
   First, they say that BGI-1's lower bound proof is incorrect. Then in this result, they prove that, in fact, in the particular counterexample networks used in BGI-1, there is a sublinear (log n) algorithm, for Model A.
5. (Kowalski-Pelc) An upper bound on deterministic broadcast.
   They generalize the algorithm above to get sublinear time algorithms for all graphs of small max distance D. They prove this in two stages: first, for $D = 2$, and then for any case where $D = o(\log \log n)$. The second of these is technical enough that we will cover the first only. It seems to contain the key ideas.
   Open question: Can we get a sublinear time deterministic algorithm for all graphs with sublinear max-distance?
6. (Kowalski-Pelc) A lower bound on randomized broadcast.
   They construct a new class of graphs, of max distance 2, in which every deterministic broadcasting algorithm requires time $\Omega(n^{1/4})$. When combined with known upper bounds for randomized algorithms (like that of BGI-1), this yields the desired exponential gap between randomized and deterministic broadcast algorithms.
7. (Kushilevitz-Mansour) A lower bound on randomized broadcast.
   They show an $\Omega(D \log(n/D))$ lower bound on the expected time to complete broadcast, for randomized algorithms.

2 Model definitions

2.1 Model A

The most important model is the deterministic model defined in BGI-1, called Model A in the subsequent errata sheet. In Model A we assume an undirected graph $G = (V, E)$ with a distinguished source node 0. In this model:
$n =$ number of nodes.
$D =$ maximum distance in the graph (number of hops) between node 0 and any other node.
$N_u =$ neighbors of u in G.

We consider a single message, which starts out at the source and has to get communicated to all nodes. We assume time slots, synchronized everywhere, that start at 0. In each slot, each node chooses whether it will be a transmitter, a receiver, or nothing (inactive). We assume that the edges indicate exactly who is within receiving range. The message receiving and collision rules are as follows:

- If node u is a receiver in slot k and exactly one of process u's neighbors transmits in slot k, then u receives the message.
- If node u is a receiver in slot k, and either 0 or > 1 of its neighbors transmits, then u receives a "null message" (nothing). Note that this is weaker than what was assumed by Gallager: he assumed the receivers got (0,1,c) information—c is different from 0. In Model A, a receiver cannot distinguish a collision from "nothing sent".
- If node i isn't a receiver in slot k, it receives nothing.

In Model A, all node programs are deterministic and identical (uniform program) except that they know their own uids and those of their neighbors. Nodes know a "close" upper bound N on the number of nodes in the network.

2.2 Model B

This is the same as Model A, except for the collision rule. Now if node u is a receiver in slot k, and either 0 or > 1 of its neighbors transmits, then u either receives a "null message" (as before) or else receives one of the actual messages sent. The choice of which happens is not under the control of the algorithm—either might happen. We say that the choice is under control of an "adversary" to the algorithm. An algorithm would have to cope with either possibility.

2.3 Randomized model

This is similar to Model A but the processes can make random choices of what to do next. Also, the processes know ϵ in case they are supposed to achieve delivery with probability $1 - \epsilon$.

In BGI-1, the processes don't have unique ids, and don't know the identities or exact number of their neighbors. However, they do know Δ, an upper bound on the maximum node degree. In Kushilevitz-Mansour, both uniform and unique processors are considered.

3 BGI-1

Bar-Yehuda, Goldreich, Itai. On the Time-Complexity of Broadcast in Multi-Hop Radio Networks

3.1 Randomized algorithm

Assume the randomized version of Model A, with knowledge of Δ (upper bound on degree).

3.1.1 Decay

The protocol is based on a Decay subroutine, which is supposed to randomly eliminate half of the contending neighbors of a node at each time slot:

```
Decay(k,m), $k \geq 1$
$coin := 1$
$counter := k$
Repeat while $coin = 1$ and $counter > 0$:
–Transmit $m$
–$count := count - 1$
–$coin := 0$ or 1, with equal probability
```

Thus, everyone chooses randomly whether to continue transmitting or not, at each step. This results in the number of transmitters halving approximately at each step. Now consider the situation where d nodes execute Decay(k,m) in synch, starting with slot 1. Define $P(k,d)$ to be the probability that, at some slot $\leq k$, exactly one of the nodes transmits. Also, define $P(\infty, d)$ to be $lim_{k\to\infty} P(k,d)$; this limit must exist, because the probabilities are nondecreasing in the first argument k (as we execute more attempts).

We can see that $P(k,0) = 0$ and $P(k,1) = 1$ (one neighbor transmits in slot 1). Theorem 1 gives bounds on $P(k,d)$ for $d \geq 2$:

Theorem 1: Let u be a node in V. Suppose that $d \geq 2$. Then:
(i) $P(\infty, d) \geq 2/3$.
(ii) For $k \geq 2\lceil \log d \rceil$, $P(k,d) > 1/2$.

Proof of (i):
They set up a recurrence:
$P(\infty, d) = \Sigma_{i=0}^{d} \binom{d}{i} 2^{-d} P(\infty, i)$.
This is just summing over all possibilities i, the probability that exactly i remain after one decay step, multiplied by the probability of success if i remain.

Separating out a few of the terms, we get:
$P(\infty, d) = \binom{d}{0} 2^{-d} P(\infty, 0) + \Sigma_{i=1}^{d-1} \binom{d}{i} 2^{-d} P(\infty, i) + \binom{d}{d} 2^{-d} P(\infty, d)$.
Now, the first term = 0, so omit it.
Then multiply both sides by 2^d:
$P(\infty, d)(2^d) = \Sigma_{i=1}^{d-1} \binom{d}{i} P(\infty, i) + \binom{d}{d} P(\infty, d)$.

Simplify the last term and subtract it from both sides:
$P(\infty, d)(2^d - 1) = \Sigma_{i=1}^{d-1} \binom{d}{i} P(\infty, i)$

Then this immediately yields the base case, for $d = 2$:
$P(\infty, 2)(3) = \Sigma_{i=1}^{1} \binom{2}{i} P(\infty, i) = 2$, so $P(\infty, 2) = 2/3$.

It also yields the inductive step: Consider $d > 2$ and assume the result for everything $< d$.
We have $P(\infty, d)(2^d - 1) = \Sigma_{i=1}^{d-1} \binom{d}{i} P(\infty, i)$,
which is equal to $\binom{d}{i} P(\infty, 1) + \Sigma_{i=2}^{d-1} \binom{d}{i} P(\infty, i)$,
or $d + \Sigma_{i=2}^{d-1} \binom{d}{i} P(\infty, i)$.

By induction, each of the P terms in the sum is $\geq 2/3$, so we have $P(\infty, d)(2^d - 1) \geq d + 2/3 \Sigma_{i=2}^{d-1} \binom{d}{i}$.

Now, this summation is just $2^d - 1 - 1 - d$, so the RHS is $\geq d + (2/3)(2^d - 1) - (2/3)(1 + d)$. It is enough to show that this is $\geq (2/3)(2^d - 1)$. That is, we want to show that $d \geq (2/3)(1 + d)$, or in other words, $3d \geq 2(1 + d)$. Having $d \geq 2$ makes this work out.

Proof of (ii):
They claim that values up to $d = 5$ hold "by inspection". That's a matter of calculation—it would be nice if they included them. I checked $d = 2$, not the others.

There is a separate case for $d \geq 6$:
They write $P(k, d) \geq P(\infty, d) - d * 2^k$. This follows from the observation that, if the node ever gets a message and all the neighbors transmit in no more than k slots, then the node must get the message within k slots. Thus, the probability of success within k slots is at least as great as the probability of success ever, minus the probability that at least one of the d neighbors performs more than k slots. The subtracted term is an overestimate for the probability that at least one of the d neighbors performs more than k slots.

3.1.2 The broadcast protocol

Decay is a nice primitive; how do we use it in a multi-hop broadcast protocol?
Also, note that the probability of success for Decay is only guaranteed to be $\geq 2/3$ if run forever, $> 1/2$ if run in "truncated" form—for $2\lceil \log d \rceil$ rounds. How do we ensure a higher probability, e.g., $1 - \epsilon$ for an arbitrary (known) ϵ?

The idea of the Broadcast protocol is that each node runs the truncated form multiple times, where the number of times depends on ϵ—specifically, $t = 2\lceil \log(N/\epsilon) \rceil$, where N is a (known) upper bound on the number of nodes. Each time it runs, it uses $k = 2\lceil \log(\Delta) \rceil$ slots, where (recall) Δ is an upper bound on its number of neighbors.

Specifically, the nodes group their slots into synchronized batches of $k = 2\lceil \log(\Delta) \rceil$. Each node waits until it first receives the source message m. Then, starting from the beginning of the next synchronized batch of slots, it executes Decay(k,m) exactly t times, each time truncated at $k = 2\lceil \Delta \rceil$. The idea is that, if one attempt succeeds with probability $> 1/2$, enough repeated attempts will succeed with as high a probability as we want.

They break their correctness proof into two lemmas.
Lemma 2: The probability that all nodes eventually receive the message is $\geq 1 - \epsilon$.

That is, with high probability, the communication doesn't die out before all processors receive the message. This lemma doesn't say when the message arrives everywhere, though. The proof is straightforward, based on Theorem 1 part (ii), which for some reason they don't cite. The idea is that, if this fails, it must be because there is some node u who doesn't receive the message, but at least one of its neighbors does. This is at most n times the probability that any particular node u doesn't receive the message but one of its neighbors does, so focus on one node u.

Start counting from the time when a neighbor of u first receives the message. From that point, there are at least t attempts by some neighbors to transmit using Decay(k,m); these must all fail since u doesn't receive the message. Each attempt succeeds with probability $> 1/2$, and the attempts are independent, so the probability of failure is $< (1/2)^t$, which is $\leq (1/2)^{\log(N/\epsilon)} = \epsilon/N$. Multiplying by n, the actual number of nodes, we get $\epsilon(n/N) \leq \epsilon$. QED Lemma 2.

Lemma 2 just gives a bound on the probability that everyone *ever* gets the message. Now, they want a time bound. An obvious upper bound is $O(D \times \log(\Delta) \times \log(N/\epsilon))$: The D arises because we need to bound the time for each "layer" of nodes (based on number of hops from the source) to receive the message, after the time for the previous layer. The rest of the expression comes from the two parameters of the Broadcast procedure.

However, they want a better bound, so they define (and here the calculations get a bit involved): $M = \sqrt{\log(n/\epsilon)}$, and $T = 2D + 5M \max(\sqrt{D}, M)$.
They use T in stating Lemma 3 (see below). What does T mean? Well, T has two terms: the first is intended to capture delays due to the diameter and the second captures delays due to conflicts. If D is small (which is a case they emphasize), then T is $O(M^2) = O(\log(n/\epsilon))$.

Lemma 3 talks about a modified version of the Broadcast algorithm, which continues forever instead of being stopped after a fixed number of iterations. For this modified version, they get two bounds:
(i) For each node u, the probability that u doesn't receive the message by time $2\lceil\log(\Delta)\rceil \times T$ is $< \epsilon/n$.
(ii) The probability that *some* node doesn't receive the message by time $2\lceil\log(\Delta)\rceil \times T$ is $< \epsilon$.

So, they have a time bound for everyone to get the message in the non-stopping version of the algorithm (with high probability). And (Lemma 2) they know that, in the real (stopping) version, *eventually* everyone does receive the message, with high probability. Now they want to combine these two facts to get a theorem giving a time bound for the real version:

Theorem 4: With high probability (here, $1 - 2\epsilon$), all the nodes receive the message by time $2\lceil\log(\Delta)\rceil T$, and all terminate within an additional time $4\lceil\log(\Delta)\rceil\lceil\log(N/\epsilon)\rceil$. The proof is not given. They say they are combining Lemmas 2 and 3, but it is unclear exactly how this works. They are adding the probabilities of not achieving the desired goal, but it seems that the time for nodes to receive m might be much better in the non-stopping algorithm than in the real one.

The presented broadcast protocol has some nice additional properties. It is simple, fast, and local. There is low message complexity, it can adapt easily to changing topologies, and it is resilient to failures. It also works in directed graph networks.

3.1.3 Other Applications of Decay

Decay can also be used to perform breadth-first search: You can determine minimum-hop routing paths, where each node learns its parent in the overall BFS tree. They use Decay, but slow it down enough so that they are unlikely to get anomalies—the distances get determined "layer-by-layer", with high probability.

Decay also helps in leader election. BGI came up with an algorithm using Decay directly, but they decided that they could generalize the idea. We can consider this as a special case of emulation of single-hop network with collision detection, on multi-hop network without collision detection. That's the subject of BGI-2, which we will see shortly.

3.2 Lower Bound for Deterministic Algorithms

This lower bound in the paper is correct, but for Model B instead of Model A. The difference is that in Model B if node u is a receiver in slot k, and > 1 of its neighbors transmits, then u either receives a "null message" (as before) or else receives one of the actual messages sent. The choice is up to an "adversary" to the algorithm, so an algorithm would have to cope with either possibility. However, a lower bound could take advantage of either possibility.

3.2.1 The graphs

The lower bound is proved with respect to a particular collection $\{\mathcal{C}_n : n \in N\}$ of sets of networks, where each graph in $\mathcal{C}_n$ has exactly $n + 2$ nodes, called $\{0, 1, \ldots, n\}$. See figure 1.

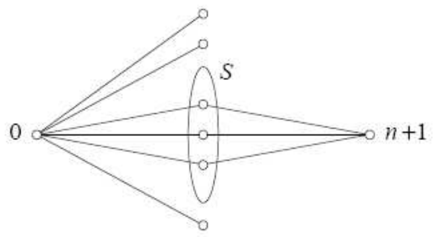

Figure 1: BGI-1 graph for randomized lower bound

The nodes of any graph in $\mathcal{C}_n$ are arranged into 3 layers, with the source 0 alone in Layer 0, the sink $n + 1$ alone in Layer 2, and the remaining nodes in Layer 1.

Let S be any nonempty subset of $\{1, \ldots, n\}$. Each graph in $\mathcal{C}_n$ has the following connections:
The source is connected to all the L_1 nodes.
Some nonempty subset of the L_1 nodes are connected to the sink.
If this subset is S, we denote the resulting graph by G_S (n is implicit here).

In these special kinds of graphs, the problem reduces to reaching the sink—the only node that won't be reached by an initial broadcast by 0. An algorithm must ensure that only one element of S sends at some time, so that its message is guaranteed to reach $n + 1$. Conversely, in a lower bound proof, an adversary will try to prevent this from occurring.

3.2.2 Reduction to Abstract Broadcast Protocols

BGI reduce the problem to a simple combinatorial game called the *Hitting Game*. Reduction is done in 3 stages, the first two of which involve restrictions on the model and generalizations of the problem statement. The third stage, in the next section, reduces the problem to the Hitting Game.

Stage 1: Restricted Broadcast protocols:
In every time slot, exactly one of source, sink is active.
Lemma 5: WLOG, we can assume the broadcast protocol is Restricted, with only a factor of 2 increase in time bound.

Stage 2: Abstract Broadcast protocols:
They add some more restrictions—see Definition 4:
1. In each slot, only nodes in L_1 are transmitters, and either the source or sink, but not both, are receivers. Messages consist merely of the transmitter's id and its S-indicator (a bit saying whether it's in S).

3. At the end of each slot all nodes in L_1 know whether someone has successfully transmitted at that round, and they even know the contents of the received message. (A strong assumption—they get immediate feedback. But note that it is OK to make strong assumptions while proving a lower bound—that just makes the lower bound result stronger.)

5. The broadcast is completed once someone in S manages to transmit alone. But this is over-optimistic: it doesn't really mean that the message has gotten to the sink, because this could be a

slot at which the source is acting as a receiver but not the sink. In this case, all the node in S has managed to do is to send the message back to the source! Restrictions 1 and 3 can be viewed as strengthening the model. Restriction 5 can be viewed as weakening the requirements—either the message reaches the sink or gets reflected back successfully to the source.

Lemma 6: If there is a Restricted Broadcast protocol that terminates within k slots, on every network in $\mathcal{C}_n$, then there is an Abstract Broadcast protocol that terminates within the same number k of slots.

They talk about how an abstract protocol executes. This is used in Lemma 7, which shows how to transform an Abstract protocol into an instance of the Hitting Game. Define $H_k = P_1, \ldots, P_k$, the history of what happened through k rounds. Here, each P_i is either the id of the unique node that transmits alone at slot i, or else is $\perp$, a special "unsuccessful" indicator. Each node's decision of whether to transmit is a function of its own id, a bit saying whether it is in S or not, and the global history so far. Express this decision as a predicate π.
Notation:
T_i = set of transmitters in slot i.
T_i^1 = set of nodes in L_1 that would transmit at round i if they were in S (for the given history).
T_i^0 = set of nodes in L_1 that would transmit if they were not in S.
The last two of these are defined in terms of the predicate π, e.g., T_i^1 = the nodes u for which $\pi(u, 1, H_{i-1}) = 1$.
Here, the second argument says that the node is in S.

3.2.3 Reduction to the Hitting Game

Definition 5: The n^{th} hitting game, played on graphs in $\mathcal{C}_n$ between Explorer and Referee. The Referee simply picks a subset $S \subseteq \{1, \ldots, n\}$. The Explorer, without knowing S, chooses sets $M_1, M_2, \ldots$. It is trying to "hit"—isolate—an element of S. The Explorer's subsequent sets M_i can depend on results of prior moves.

What happens at move i:
If $|M_i \cap S| = 1$, the game is over, and Explorer is said to win at round i.
If $|M_i \cap \bar{S}| = 1$, the game continues, but the Referee reveals the unique element in the intersection to the Explorer.
Otherwise, the Explorer learns nothing.

The relationship between this and the original problem is not exactly transparent. The first condition is supposed to capture the situation where just one member of S transmits, which is how the sink learns the source's message. The second condition is less clear: it indicates at least the potential that only one node broadcasts—since the decision to broadcast can depend on the S-indicator. Of course, broadcasts for nodes not in S don't really reach the sink, but remember that in the abstract protocol, we count success when anyone succeeds in transmitting alone, whether connected to the sink or not!

Lemma 7: If there is an Abstract Broadcast protocol that terminates within k slots, on every network in $\mathcal{C}_n$, then there is a $2k$-move winning strategy (sequence of moves $M_1, \ldots, M_{2k}$ that always wins, regardless of the choice of S) of the n^{th} Hitting Game.

The (appendicized) proof is wrong: Lemma 7 is incorrect for the given Model A.

3.2.4 Lower Bound on Hitting Games

They prove a linear lower bound on the number of moves to win the n^{th} Hitting Game (this must work in the worse case, for all S). Specifically, they define an "adversary" that constructs a set S that "fools" a particular sequence $M_1, M_2, ..., M_t$ of moves. This is making a bit of a jump—the problem was defined in terms of an adaptive choice of moves M_i, whereas the lower bound is proved only for oblivious sequences of moves. It is at least theoretically possible that this could make a difference: that there is an adversary that fools all oblivious strategies, but some adaptive strategy could beat it!

However, they argue—some tricky reasoning—this is in the errata note, actually—that the particular adversaries they construct, which beat all oblivious strategies, will also beat all adaptive strategies. This is because the adversaries are defined in a way that determines all answers from the moves themselves: all non-singletons are answered with $\perp$, and all singletons are answered with the same element (which is declared not to be in S). So, if we take any arbitrary game strategy that always does well, and consider the moves it takes when all answers are as above, we get a fixed (oblivious) strategy that also does well against the given adversaries. But this is what they prove is impossible.

So we construct the set S for a given fixed sequence $M_1, \ldots, M_t$ of moves. Construction is described by simple code, top of p. 23. Start with $S = \{1, \ldots, n\}$. While there exist i, x such that $M_i \cap S = \{x\}$ do
–Choose such i, x
–Remove x from S
–While there exists j such that $|M_j \cap S| = |M_j| - 1 > 0$ do
—Remove any element of $M_j \cap S$ from S. (this is putting a second element of M_j into $\bar{S}$.

Lemma 9: The resulting S has the right intersection properties with respect to the M_i's. Namely, for every i:
(i) $M_i \cap S$ is not a singleton.
(ii) If $M_i \cap \bar{S}$ is a singleton then M_i is a singleton.

But Lemma 9 isn't quite enough: We haven't ruled out the case that S is empty, which would not be an allowable choice for the adversary. For this, we have:

Lemma 10: If $t \leq n/2$ then output S is nonempty.
Proof: Each move M_i "causes" the removal of at most two elements from S: one for the outer while loop (to ensure that $M_i \cap S$ isn't a singleton), and one for the inner while loop (to try to ensure that $M_i \cap \bar{S}$ isn't a singleton).

Now they pull all this together to get a lower bound on the Hitting Game. Namely, consider an oblivious Explorer strategy of length $t \leq n/2$ (we argued above about why oblivious is enough), and construct set S using the given adversary. When we run the moves from the beginning, $M_1, \ldots, M_t$, with this set S, we get no winning moves (by part (i) of Lemma 9). Which means that the game doesn't finish by time t.

What goes wrong with the lower bound for the $\mathcal{C}_n$ graphs? Well, there is a trivial 3-round protocol that works if the sink can transmit spontaneously, before receiving a message. Since the sink is assumed to know the ids of its neighbors—the elements of S—it can simply broadcast the set S. Then the smallest element of S can be the one to transmit the actual source message. This is quite odd...

3.3 The Error in the Proof

Errata for BGI-1: from a web page indicated in Kowalski-Pelc, by following a pointer to a useful 5-page writeup.

3.3.1 Overview

Here, they articulate Model A and Model B, and point out the differences in collision handling. In Model A, nothing is received on a collision, the channel looks the same as when it's idle. In Model B, either one of the transmissions is received, or else nothing is received. Model B provides weaker guarantees, making it harder to build an algorithm, and easier to prove a lower bound. In fact, not just the proof in BGI-1, but the actual result—the linear lower bound for deterministic algorithms—doesn't hold for Model A—Kowalski-Pelc give a counterexample.

The error is buried in the proof of Lemma 7, which shows how to get from the Abstract model to the Hitting Game. This only appeared in the Appendix, and in too little detail to see the mistake. The proof in BGI-1 is still (apparently) correct, for Model B.

3.3.2 Technical Details

The change in assumption about collision behavior causes changes in the three models in the paper: the original, the Restricted, and the Abstract. All nodes are now guaranteed to receive a message if exactly one transmits, as before. But may also receive if more than one nodes transmit. They claim that the constructions that reduce to this point are still OK, with these changes. The problem that occurred earlier was in Lemma 7. They claim that Lemma 7 works for the revised models, though not before. The proof in BGI-1 didn't include enough details to tell, so they now fill in more details in the errata note.

What follows is a detailed Proof of Lemma 7 (for weaker collision assumptions). It describes how to use a (worst-case) t-slot broadcast strategy for the abstract model to yield a $2t$-move strategy for the Hitting Game. Each round (slot) in the protocol is used to determine two moves of the game. Referee answers are then used to determine the outcome of the communication round. The way the referee answers determine these outcomes has to be consistent with what is allowed in the Abstract Model. Before, it wasn't; now it is.

So, fix some particular abstract protocol $\mathcal{A}$. Then (recall), each node's decision of whether to transmit is a function of its own id, a bit saying whether it is in S or not, and the global history so far. This decision is expressed as a predicate π. Here, the history is a sequence of P_i, each of which is either the id of a unique node that transmitted alone at slot i or else $\perp$. Each round of the abstract protocol is turned into two moves in the game, which are described by two sets—T_i^1 and T_i^0 defined earlier. Recall T_i^1 = nodes that would send if they are in S, based on history so far; T_i^0 = nodes that would send if they are *not* in S, based on the history so far.

Now consider the Referee's responses, defined as usual: win if $M \cap S$ is a singleton, respond x if $M \cap \bar{S} = \{x\}$, otherwise $\perp$. If the game wins, we declare that abstract protocol $\mathcal{A}$ is also finished. Otherwise, we need to specify the behavior at the round in question, in $\mathcal{A}$—specifically, what messages are delivered to whom. The rule they use is:
If the second referee answer (to the query for T_i^0) is a singleton $\{x\}$, that is, if $T_i^0 \cap \bar(S) = \{x\}$, then deliver just the message from x. Otherwise deliver nothing.

Now the key point: is this delivery rule consistent with the requirements of the Abstract model? The answer is no for Model A, but yes for Model B. Why is this so for Model B? Let's examine the two cases:

Case 1: We do deliver a message (from x) in $\mathcal{A}$. We must show that we are allowed to do this, according to the rules of the abstract protocol model. This is the interesting case. The actual transmitters, in step i of $\mathcal{A}$, are $(T_i^1 \cap S) \cup (T_i^0 \cap \bar{S})$. That is, the nodes who are in S and whose rule says they would transmit if they are in S, plus the nodes that are not in S and whose rule says they would transmit if they are not in S. Note that this set does contain x, because the second set in the union does. x need not be the only transmitter, yet Model B allows this delivery anyway.

Case 2: We don't deliver any message in $\mathcal{A}$, and the broadcast is not completed. We must show we are allowed to do this, according to the rules of the abstract protocol model. In this case, it must be that:
—$T_i^1 \cap S$ is not a singleton (because the move for T_i^1 would then have won the game)
—$T_i^0 \cap \bar{S}$ is not a singleton (because then the message would have been delivered)
So, the union $(T_i^1 \cap S) \cup (T_i^0 \cap \bar{S})$ is also not a singleton. Which means Model B allows us to not deliver a message. OK, so we can translate the responses back into allowable message deliveries in $\mathcal{A}$. Now, note that, if a run of $\mathcal{A}$ completes within t rounds, the corresponding sequence of $2t$ Hitting Game moves will win. Thus, if $\mathcal{A}$ always finishes in t rounds the corresponding sequence of Hitting Game moves always (that is, for any S) finishes within $2t$ moves. QED Lemma 7.

4 BGI-2

Bar-Yehuda, Goldreich, Itai. Efficient Emulation of Single-Hop Radio network with Collision Detection on Multi-Hop Radio Network with no Collision Detection

This paper presents an efficient randomized emulation algorithm, to emulate any algorithm for a single-hop network with collision detection on a multi-hop network with no collision detection. Each step of the single-hop network is emulated by $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$ rounds of the multi-hop network, and succeeds with probability $\geq 1 - \epsilon$. They also show how to emulate polynomially many steps while maintaining a probability of failure $\leq \epsilon$.

Corollary: By emulating Willard's leader election algorithm, they show an efficient randomized algorithm for leader election in a multi-hop network.

4.1 Introduction

They aim to define different models for wireless networks, and show that, in a sense, they are all equivalent.

The first model presented is the "single-hop" model. This is the model used by a single segment of an Ethernet network. It has the following characteristics:

1. All processors share one channel.
2. Synchronous rounds (slots).
3. In each round:
    - If 1 sends, all receive it.

- If 0 send, no one receives anything (of course).
- If > 1 send, all receive special collision notification c (Collision Detection (CD))

Next, they talk about the "multi-hop" model. This is similar to the networks we've been discussing in this class. The essential characteristics are:

1. Arbitrary connected graph, as opposed to complete graph from single-hop.
2. Synchronous rounds.
3. In each round:
   - Each processor can choose to be either a transmitter or receiver.
   - A processor receives only if it's a receiver.
   - If it's a receiver, then:
     - If 1 neighbor sends, it receives the message.
     - If > 1 neighbor sends, it may receive one of the messages, detect the collision, or receive nothing. Note that this is the same as Model B presented in the BGI-1 errata. Receivers cannot distinguish between the case where nothing was sent, and the case where a collision simply resulted in silence.
4. Don't require uids.

When designing protocols for multihop networks, it can be difficult to overcome the lack of CD and the challenges of unknown topology. They suggest designing algorithms for the Ethernet model, and then emulating them to get protocols for multihop. This may not be perfectly efficient, since the emulation adds some overhead, but it may be simpler than trying to design a distributed algorithm. Depending on the efficiency requirements of your application, this methodology may well be perfectly acceptable.

The randomized emulation presented here runs in $O((D+\log\frac{n}{\epsilon})\log\Delta)$ slots. Notice that this is the time needed to emulate one round, and is essentially the same as the bound on the broadcast time from BGI-1. The emulation scheme uses the two primitives from BGI-1: *Decay* and *Broadcast*.

Recall that *Decay* allows each processor to receive, with probability $> 1/2$, a message sent by one of its neighbors, regardless of the number of neighbors that want to send it a message.

Broadcast makes use of *Decay*, and behaves exactly as described in BGI-1. The only difference here is that multiple sources may be transmitting at once. The bound given applies to the probability that everyone receives SOME message.

4.2 The Emulation of a Single Round

The emulation procedure tries to emulate one round of the Ethernet model in multiple rounds of the multi-hop model. The emulation proceeds in three phases:

1. Propagation: All initiators choose random tags and *Broadcast* the message they wish to send. Everyone remembers the first message they receive (initiators always hear their own message first)

2. Detection: The goal of this phase is to implement an abstract CD mechanism. This phase attempts to detect when there is more than 1 initiator. If there is more than one initiator, there must be two different tags in the system, since each message was associated with a randomly chosen tag. Moreover, somewhere, two different tags must appear at neighboring nodes. Basically, the nodes then compare tags with their neighbors. They do this by systematically considering successive bits of the tags. Each check takes $2 \log \Delta$ rounds. For the i-th bit of the tag, all nodes which have a 1 for that bit become transmitters, using the *Decay* primitive. All other nodes become receivers. If any node actually receives a message (ie, functioned as a receiver when some other node was transmitting), then the two nodes have different tags, and a collision occured. Any node receiving a message sets a flag indicating that a collision was detected.

3. Notification: This phase informs all nodes if a collision occured. Every node that detected a collision in the previous phase sends a *Broadcast* of some standard collision message using failure probability $\frac{\epsilon}{3}$. Any node receiving this message knows a collision occurred. All nodes receive this message with high probability.

4.2.1 Analysis

Now that we've seen the algorithm for emulation, let's try to analyze it and validate BGI's claims.

Lemma 1: Assuming that the propagation phase succeeded (everyone received at least one message):

1. If there was a single initiator, then after the detection phase, everyone has the right message and no conflicts are detected.
2. If there is more than one initiator, then with probability $\geq 1 - \frac{\epsilon}{3}$, at the end of the detection phase, some vertex has discovered the conflict.

Proof: (1) is obvious. If only one person tried to send, everyone will hear it correctly. This is the result from BGI-1.

(2) follows simply. Since the network is connected, we know that there are two adjacent nodes which heard different messages, which means that they have different tag. For every bit i, the two tags differ with probability 1/2. This means that one of the nodes will detect it with probability 1/2 at round i. In other words, *someone* detects the collision at round i with probability 1/4. Thus, the probability that a collision occurred but was not detected over the k rounds is $\leq (1 - \frac{1}{4})^k$. k is defined to be $\lceil 2.5 \log \frac{\epsilon}{3} \rceil$, which leads to the probability of a collision not being detected being $\leq \frac{\epsilon}{3}$.

Lemma 2: The entire protocol requires $(2 + o(1))B_\epsilon$ rounds to execute, where B_ϵ is defined to be $O((D + \log(\frac{n}{\epsilon})) \log \Delta)$. This simplifies to simply $O(B_\epsilon)$.

Proof: The Propagation and Notification phases simply consist of running the *Broadcast* primitive, which executes in $O(B_\epsilon)$ time as shown in BGI-1. The Detection phase consists of k iterations of *Decay*, where each iteration takes 2 log Δ rounds to complete. Since $k = \lceil 2.5 \log \frac{\epsilon}{3} \rceil$, the total time for the algorithm to complete is dominated by $O(B_\epsilon)$, and the Lemma holds.

Now we combine Lemmas 1 and 2 to get **Theorem 1**: If there is a single initiator, then with probability $\geq 1 - \frac{\epsilon}{3}$, everyone receives its message. If, however, there are more initiators, then with probability $\geq 1 - \epsilon$, everyone detects a conflict.

Proof: For any number of initiators, the Propagation phase succeeds in disseminating the message everywhere with probability $\geq 1 - \frac{\epsilon}{3}$, because it uses the *Broadcast* primitive. With more initiators, the Detection phase succeeds with probability $\geq 1 - \frac{\epsilon}{3}$. The last possible point of failure is the *Broadcast* in the Notification phase. Since this failure is bounded by $\frac{\epsilon}{3}$, the theorem holds.

Based on Theorem 1, this algorithm provides successful emulation. In other words, if possible, it delivers the message. Otherwise, it detects a collision. Of course, this is all with some high probability, not guaranteed.

4.3 A side issue: Implementing a CD mechanism in arbitrary multi-hop networks

A side issue: BGI present a method for implementing a CD mechanism for a multi-hop network directly, without turning it into a single-hop network first. The algorithm they use is as follows:

1. Any node wishing to transmit selects a random tag.
2. All transmitters perform *Decay(m,tag)* k times.
3. Any non-transmitter listens, and detects conflict if it hears 2 messages with different tags.
4. Receivers remember the first message they hear.

This can be shown to guarantee collision detection at the receiver with high probability. (But they don't actually show it...)

This method is for having receivers detect collisions. Most models assume that the *transmitters* are detecting the collisions. Essentially, they do two rounds. The first round uses their method as described above. They then have all receivers who detect collision broadcast a message. BGI claim that in the second round, the original transmitters either receive an explicit collision report, or else detect a collision. This isn't really obvious, since it might be possible not to hear anything. If two neighbors of the original transmitter both send collision reports, these could collide. Remember that the original definition mentions that a collision might not be distinguishable from no transmissions at all. BGI don't address this, and it makes me somewhat nervous, seeing as this was basically the error from the first paper. . . .

4.4 Emulating an Entire Algorithm

All they do here is choose ϵ cleverly. They let ϵ' be the probability of error for a single round. This number is then equal to $\frac{\epsilon}{t}$, where t is a known upper bound on the number of rounds. Thus, for t rounds with probability of error ϵ', the probability that the algorithm succeeds is $(1-\epsilon')^t > 1 - t\epsilon'$.

If they don't know an upper bound, they can adapt by decreasing epsilon sufficiently at each step, so that the total error probability stays bounded. The analysis here is on page 8/9 of the BGI-2 paper, together with Theorem 2 which pulls the whole thing together. Basically, Theorem 2 bounds the time taken to emulate any Ethernet algorithm.

4.5 Applications and Conclusions

As an example, they assert that Willard's Ethernet algorithm for leader election can be run in a multi-hop network. Willard's algorithm uses collision detection, and elects a leader in $O(\log \log n)$ rounds. Thus, BGI's emulation scheme leads to a $O(B_\epsilon \log \text{lon } n)$ algorithm for leader election on multi-hop networks without CD mechanisms.

Limitation of the approach: It requires global synchronization for every round of the emulated algorithm. This is a consequence of the slotted notion of time used in the algorithms. This won't be efficient for distributed algorithms with a "local" flavor, (e.g., the topology control algorithm from Li and Halpern). If global synchronization is not practical, this emulation scheme will no longer be a reasonable approach.