Contention Resolution on Multiple Channels with Collision Detection

Jeremy T. Fineman
Georgetown University
Washington, DC, USA
jf474@georgetown.edu

Calvin Newport
Georgetown University
Washington, DC, USA
cn248@georgetown.edu

Tonghe Wang
Georgetown University
Washington, DC, USA
tw473@georgetown.edu

ABSTRACT

In this paper, we consider the classical *contention resolution* problem in which an unknown subset of n possible nodes are activated and connected to a shared channel. The problem is solved in the first round that an active node transmits alone (thus breaking symmetry). Contention resolution has been an active research topic for over four decades. Accordingly, tight upper and lower bounds are known for most major model assumptions. There remains, however, an important case that is unresolved: contention resolution with multiple channels and collision detection. (Tight bounds are known for contention resolution with multiple channels, and contention resolution with collision detection, but not for the combination of *both* assumptions.) Recent work [14] proved the first non-trivial lower bound for randomized solutions to this problem in this setting. The optimality of this lower bound was left an open question. In this paper, we answer this open question by describing and analyzing new contention resolution algorithms that match, or come within a $\log\log\log n$ factor of matching, this bound for all relevant parameters. By doing so, we help advance our understanding of an important longstanding problem. Of equal importance, our solutions introduce a novel new technique in which we leverage a distributed structure we call *coalescing cohorts* to simulate a well-known parallel search strategy from the structured PRAM CREW model [16] in our unstructured distributed model. We conjecture that this approach is relevant to many problems in the increasingly important setting of distributed computation using multiple shared channels.

Keywords

contention resolution; collision detection; symmetry breaking

1. INTRODUCTION

The contention resolution problem assumes an unknown subset of n possible nodes that are *activated* and connected to a shared multiple-access channel (MAC). The problem is solved in the first round that one of these *active* nodes transmits alone on the shared channel. Contention resolution (which is also sometimes called *leader election* or *wake-up*, depending on the model assumptions and context) is one of the longest studied problems in the theory of distributed algorithms. It was first identified in the 1970 paper on the ALOHA radio network and variations have been extensively explored by theoretical computer scientists and mathematicians ever since; e.g., [15, 12, 10, 13, 9, 5, 2, 8, 11, 4, 14]. In this paper, we advance an important open case regarding this problem, and in doing so also introduce a novel algorithmic technique potentially applicable to many other problems in this setting.

An Open Problem. Contention resolution has been studied under many different model variations. Two of the most important assumptions concern collision detection and number of available channels. Tight bounds are known for contention resolution on a single channel with and without collision detection [5, 11, 6, 14], and for contention resolution on multiple channels without collision detection [3, 14]. (See Section 2 for details). Until recently, however, little was known about the remaining case of multiple channels with collision detection. In 2014 [14], progress was made with a new lower bound that established $\Omega\left(\frac{\log n}{\log \mathcal{C}} + \log\log n\right)$ rounds are needed to solve contention resolution with collision detection and $\mathcal{C} \geq 1$ channels (with high probability in n). This bound holds *even* if we consider the easier restricted case where only two nodes out of the n possible will be activated. In [14], the question of this bound's optimality was left as an open problem—though it was suspected to be *not tight* because it followed from the limits of a specific proof technique and had no connection to a particular upper bound intuition.

Our Solution. In this paper, we resolve this open problem. In more detail, we show (perhaps surprisingly) that the lower bound from [14] *is tight* or within a $\log\log\log n$ factor of tight for all relevant values of n and $\mathcal{C}$. To do so, we describe and analyze two new contention resolution algorithms for MACs with collision detection and $\mathcal{C} > 1$ channels. The first algorithm (presented in Section 4) solves the problem in an optimal $O\left(\frac{\log n}{\log \mathcal{C}} + \log\log n\right)$ rounds in the restricted case where only two nodes are activated—exactly matching the lower bound from [14], which applies to this restricted case. The second algorithm (presented in

PODC'16, July 25–28, 2016, Chicago, IL, USA
© 2016 ACM. ISBN 0-89791-88-6/97/05...$15.00
DOI: http://dx.doi.org/10.1145/2933057.2933110

Section 5) solves the problem in a near optimal $O\Big(\frac{\log n}{\log \mathcal{C}} + (\log\log n)(\log\log\log n)\Big)$ rounds for the general case where any number of nodes are activated—coming within a $\log\log\log n$ factor of the existing bound for large $\mathcal{C}$ (i.e., when the $\log\log n$ term dominates) and otherwise matching the bound.

Our Techniques. Our two-node algorithm works in two phases. The first phase renames the two active nodes with unique ids from the range $\{1, 2, ..., \mathcal{C}\}$ instead of $\{1, 2, ..., n\}$ (for the case where $\mathcal{C} > n$, we use only the first n channels). Notice, with a single channel, renaming (with high probability in n) would require $\Omega(\log n)$ rounds, as it takes this many rounds to even just break symmetry between two nodes with respect to their transmit/receive behavior. In our setting, however, we can leverage the presence of $\mathcal{C}$ channels and collision detection to speed up this process to require only $O(\log n/\log \mathcal{C})$ rounds (with high probability).

Once the two nodes have renamed themselves in a smaller id space, we consider the binary tree with the leaves labeled with this smaller set of $\mathcal{C}$ possible ids. Let i and j be the unique ids of the renamed nodes. Let P_i and P_j be the two paths in our tree from the root to the leaves labelled i and j, respectively. In our second phase, we associate a channel with each node in this tree. We use these channels to perform a binary search over the tree's $O(\log \mathcal{C})$ levels to identify the level where P_i and P_j first diverge. Once we identify this key divergence, the node associated with the path that splits to the *left* (by some fixed labeling of the child orientations) is the *winner* and can go transmit alone on the primary channel[1] to solve the problem. This second stage requires $O(\log\log \mathcal{C}) = O(\log\log n)$ rounds and is deterministic.

Our general algorithm, which works for any number of nodes, builds on a similar structure, but requires more complex analysis and the introduction of novel algorithmic strategies to overcome the issues introduced by larger sets of active nodes. In more detail, this general algorithm deploys three steps. The first step reduces the number of active nodes to a count in $O(\log n)$ in $O(\log\log n)$ rounds using an aggressive knock out strategy. This preliminary reduction serves to simplify analysis in the remaining steps.

The second step renames the remaining nodes with unique IDs from the space $[\mathcal{C}/2]$. As with the two node algorithm, this renaming leverages the presence of multiple channels. Each active node chooses a channel from 1 to $\mathcal{C}/2$ with uniform randomness, then transmits using its collision detector to determine if it is alone. If *any* node is alone on its channel, it can adopt the channel label as its unique ID and notify the nodes that have not yet adopted a new ID to become inactive. To handle the case where $\mathcal{C}$ is small compared to $\log n$, we interleave these renaming attempts with continued attempts to reduce the active nodes using a knock out strategy. Analysis of the reduction rounds in this phase is complicated by the fact that we need to achieve reductions to active node counts (potentially) much smaller than $\log n$ with high probability in n. The analysis of the renaming attempts deploys a custom balls-in-bins argument that treats the nodes as balls and the channels they choose as bins.

[1]The definition of contention resolution with multiple channels fixes one channel to play the role of the *primary* channel. The problem is solved when an active node broadcasts alone on the primary channel.

When the renaming completes, we proceed to the third step which elects a leader and features the primary algorithmic contributions of this paper. For this step, we once again consider a binary tree with one leaf for each possible id in our reduced space, and focus on the paths from the root to the *occupied* leaves corresponding to the ids of active nodes. As in the two node algorithm, in $O(\log\log n)$ rounds we can complete a binary search that finds the level closest to the root in which all paths occupy unique tree nodes. Also as in the two node algorithm, we can use the identification of this level to reduce the number of active nodes by a factor of at least 2. After our first step, we have at most $O(\log n)$ nodes, so we would need $O(\log\log n)$ such searches. The resulting time complexity for this strategy, however, is too large. To optimize this process to our desired $(\log\log n)(\log\log\log n)$ rounds, we need to speed up the time required by these searches.

To accomplish this speed up, we introduce a new technique called ***coalescing cohorts***. The idea is that when we first enter this step, the active nodes are not coordinated. As we proceed with our basic strategy of conducting searches to reduce active nodes, however, we *coalesce* coordinated groups of nodes that we call *cohorts*. A key guarantee of these cohorts is that after each search, they each have the same size p and their members have unique IDs from a known set of size p. (We cannot immediately solve leader election from this property, however, because the total number of cohorts in the system is unknown.) This structure allows us to simulate a CREW PRAM parallel search algorithm [16] that has p processes work in parallel to perform a $(p+1)$-ary search. After each search, this cohort size p doubles, and therefore the search time decreases rapidly enough that the sum of the rounds spent to conduct $O(\log\log n)$ search/reductions is in $O\Big((\log\log n)(\log\log\log n)\Big)$.

Impact. These results generate both specific and general impacts. The specific impact is that they close a long-standing open question regarding a classical distributed algorithm problem. The general impact comes from the new techniques we developed to prove our results. In particular, our *coalescing cohorts* strategy provides a method to quickly build large collections of coordinated nodes in distributed multiple channel models. In this paper, we use the structure in these groups to simulate efficient strategies from the parallel algorithms literature. We conjecture that this strategy can be combined with a variety of well-known parallel algorithms to speed up computation in our distributed model. Even without parallel algorithm simulation, however, we note that the structure provided by these cohorts still provides a powerful algorithmic tool that can potentially be leveraged in developing efficient solutions for many problems.

2. RELATED WORK

The study of contention resolution developed out of the network random access method introduced in the ALOHA paper in 1970 [1]. Early work on this problem focused on the *stability* of contention resolution algorithms for different packet arrival rates; e.g., [15, 12, 10] (see [7] for a good survey). By the mid-1980's, however, the literature split, with an increasingly number of mathematicians and computer scientists studying a one-shot version of the problem where a single set of packets arrives at the beginning of the execution, and the problem is solved once the first packet is successfully delivered e.g., [13, 9, 5]. This is the version of the problem we study in this paper.

As mentioned, contention resolution has been considered under many model assumptions. The two assumptions relevant to this paper are collision detection capability and the number of available channels. Many of the original papers on contention resolution (e.g., [13, 9, 5]) assume *collision detection*, which they define (as do we) such that *all* nodes learn of a collision during any round with two or more transmitters.[2] A straightforward algorithm solves contention resolution in $O(\log n)$ rounds in this setting with probability 1: active nodes use collisions to guide a descent through a binary search tree over the n possible ids to identify the smallest id of an active node. This solution is optimal for solving this problem with high probability in n [14]. Without collision detection, by contrast, contention resolution can be solved with high probability in $O(\log^2 n)$ rounds. Jurdzinkski et al. [11] proved this near optimal for uniform[3] algorithms and Colton et al. [6] eliminated the remaining gap. Recently, Newport [14] proved this bound optimal for all algorithms.

Providing nodes additional channels also speeds up contention resolution. Daum et al. [3] show how to solve contention resolution with high probability in $O\Big(\log^2 n/\mathcal{C} + \log n\Big)$ rounds with $\mathcal{C} \geq 1$ channels—achieving a linear speed up in $\mathcal{C}$ until reaching the lower limit of $\Omega(\log n)$. In [3], this bound was proved near tight for uniform algorithms. Newport [14] subsequently proved it tight for all algorithms. This same paper proved that solving contention resolution with high probability with collision detection and multiple channels requires $\Omega\Big(\frac{\log n}{\log \mathcal{C}} + \log\log n\Big)$ rounds for all algorithms, even if we consider the restricted case where only two nodes are activated. At the time, the best known upper bound for these assumptions required $O(\log n)$ rounds (i.e., the optimal algorithm for collision detection on a single channel). Finally, we note the parallel binary search strategy we leverage in our new upper bound for general network size came from Snir's classic 1985 paper on parallel searching [16].

3. THE CONTENTION RESOLUTION PROBLEM

We parameterize our model with integers $\mathcal{C} > 0$ and $n \geq 2$. We assume a set V consisting of n *nodes* each executing a randomized algorithm, and a collection of $\mathcal{C}$ multiple-access channels (called *channels* for short in the following), labelled $1, 2, ..., \mathcal{C}$. Our algorithms do not require nodes to have unique ids (though the lower bounds in [14] hold even if they do).

At the beginning of an execution, some arbitrary subset $A \subset V$ of the n possible nodes are designated as *active.* (When specified, we sometimes consider a *restricted* case in which $|A|$ must equal 2.) The active nodes are the only nodes that participate in the execution. Time then proceeds in synchronous rounds. In each round, each node in A must make two decisions: (1) it must choose a single channel from 1 to $\mathcal{C}$ on which to participate; and (2) it must decide whether to *transmit* a message or *receive.* Each individual channel behaves like a standard MAC with (strong) collision detection. In more detail, fix some node $u \in A$ that chooses to participate on channel i in round r. If no node transmits on i in this round, u detects silence. If exactly one node transmits on i in this round, u receives this message. If two or more nodes transmit on i, u receives a collision notification. Notice, as in [14], we assume the classical definition of collision detection (common in much of the original work on this problem), where both transmitters and receivers learn about message collisions on their channel in a given round.

The *contention resolution* problem is solved in the first round in which a single node transmits on channel 1. We assume all nodes start during the same round. It is easy to transform a solution that works in this model to a solution that works in the harder model where nodes can start during different rounds, at the cost of a factor of 2 in time complexity. In nonsimultaneous wake-up case, we can have each node listen for two rounds on channel 1. If both rounds are silent, it starts running a modified version of the protocol where node broadcasts in the odd rounds (on channel 1) and runs the protocol in the even. If the node instead hears a collision or message in the first two rounds, it just stop participating in the algorithm.

In this paper, we consider results that hold with high probability in n (the maximum number of nodes that might be activated), which we define to mean probability at least $1 - 1/n^c$ for some fixed constant $c \geq 1$. We define the notation $[i, j]$, for integers $i \leq j$, to denote the range $\{i, i+1, ..., j\}$, and define $[i]$, for integer $i > 0$, to denote $[1, i]$.

4. CONTENTION RESOLUTION FOR TWO NODES

We begin by considering the restricted case where exactly two nodes are activated. Under this assumption, we describe and analyze an algorithm that solves contention resolution in $O\Big(\frac{\log n}{\log \mathcal{C}} + \log\log n\Big)$ rounds, with high probability. This algorithm matches the lower bound from [14], which holds for this two node assumption. This algorithm is also useful as it isolates and highlights some of the general ideas leveraged by the more involved general algorithm that follows.

In the following we assume that $\mathcal{C}$ is a power of 2 (the strategies are easily modified to handle other values). We also assume $\mathcal{C} \leq n$. Notice, no optimality is lost by this latter assumption.[4]

The TwoActive Algorithm. The algorithm executes in two steps. The first step implements an ID reduction strategy that renames the two active nodes with unique ids from the range $[\mathcal{C}]$. The two nodes each choose a channel from the $\mathcal{C}$ available channels and then transmit and use their

[2]In more recent years, some papers also assume *receiver collision detection*, in which the transmitters involved in the collision do not directly learn about the collision (as would be the case with half-duplex transmitters).

[3]This is one of several terms used to describe a restriction common in many contention resolution lower bounds: the transmission probabilities used by the algorithm are fixed at each node in advance. This restriction forbids, for example, nodes to base their transmission probability in one round on coin flips from an earlier round.

[4]The lower bound for $\mathcal{C} > n$ is $\Omega(\log\log n)$, and our algorithm executes in $O(\log\log n)$ rounds when run on n channels.

```
TwoActive (for node $i \in A$)
(Step #1: ID Reduction)
  do
    Choose $ID_i$ from $[1, \mathcal{C}]$ with uniform
      randomness
    Transmit on channel $ID_i$
  until $i$ is alone on channel $ID_i$

(Step #2: Symmetry Breaking)
  $L \leftarrow$ SplitCheck$(0, \lg \mathcal{C}, ID_i)$
  if the node at level $L$ in $T_{\mathcal{C}}$ on $P_i$ is left child
    of parent at level $L-1$
  then transmit on channel 1
SplitCheck$(l, r, id)$
(return the level where $P_i$ and $P_j$ split)
if $l \geq r$, then,
  return $l$
else
  $m = \lfloor (l+r)/2 \rfloor$
  Transmit on channel $\lceil id/2^{\lg \mathcal{C} - m} \rceil$
  if collision is detected then
    return SplitCheck$(m+1, r, id)$
  else
    return SplitCheck$(l, m, id)$
```

Figure 1: The TwoActive Algorithm

collision detectors to see if they are alone. (This step leverages the strong collision detection assumption that allows transmitters to detect collisions.) The nodes repeat this random channel selection strategy until they detect they are on unique channels: at which point, each node then adopts the label of its selected channel as its new unique id. This step requires $O(\log n / \log \mathcal{C})$ rounds (with high probability).

The second step makes use of these new IDs to efficiently break symmetry among the two nodes. To do so, the nodes consider a canonical full binary tree $T_{\mathcal{C}}$, with $\mathcal{C}$ leaves labelled with the values in $[\mathcal{C}]$. Note that $T_{\mathcal{C}}$ has a height of $h = \lg \mathcal{C}$. Let i and j be the two active nodes, and let ID_i and ID_j be their ids selected in the previous step. Node i considers the unique simple path in $T_{\mathcal{C}}$ from the root to the leaf labelled ID_i, and j does the same with respect to ID_j. We call these paths P_i and P_j, respectively. Notice that these paths begin together at the root and split at some point by the time they reach their respective leaves. This second step conducts a binary search over the h levels of the tree to find the smallest level at which the two paths diverge. This search requires an assignment of a unique channel to each node at the tree level being checked at a given step of the algorithm; that is, the multiple channel assumption is necessary for this step as well. Once we have found this level—which we call ℓ in this exposition—breaking symmetry is simple: the single node that is a left child of its parent at $\ell - 1$ (on its path from the root) *wins* and transmits on channel 1 to solve the problem. This step requires $O(\log h) = O(\log \log \mathcal{C}) = O(\log \log n)$ rounds.

Analysis. We are now ready to state our main correctness theorem:

Theorem 1. *In a network with* 2 *active nodes (out of n possible nodes) and $\mathcal{C}$ channels,* TwoActive *solves contention resolution in $O\Big(\frac{\log n}{\log \mathcal{C}} + \log \log n\Big)$ rounds, with high probability.*

The correctness of our theorem is a direct consequence of the following two lemmas.

Lemma 2. *With high probability: nodes i and j conclude Step #1 with $ID_i \neq ID_j$, during the same round $r \in O\Big(\frac{\log n}{\log \mathcal{C}}\Big)$.*

Proof. Fix $t = \lg n / \lg \mathcal{C}$. The probability that nodes i and j select the same channel for t consecutive rounds in the ID reduction phase is bounded as:

$$\prod_{r=1}^{t} Pr\{ID_i = ID_j \text{ for round } r\} = \prod_{r=1}^{t} (1/\mathcal{C})$$

$$= (2^{-\lg \mathcal{C}})^t = 2^{-\lg n} = \tfrac{1}{n},$$

which is sufficiently small for our lemma statement. (Notice, we can easily adjust this result for probability $1/n^c$, for constant $c > 1$, by simply increasing t by a factor of c.) □

Lemma 3. *Assuming that nodes i and j begin Step #2 during the same round with $ID_i \neq ID_j$, then this step will go on to solve contention resolution in an additional $O(\log \log n)$ rounds.*

Proof. Consider a binary array $B \in \{0,1\}^{\lg \mathcal{C}+1}$, defined such that for each $m \in [0, \lg \mathcal{C}]$, $B[m] = 1$ iff P_i and P_j pass through the same node in level m of $T_{\mathcal{C}}$. Notice that $B[0] = 1$, $B[\lg \mathcal{C}] = 0$, and the array, when read from small to large indices starts with 1's then changes over to 0's. The SplitCheck subroutine implements a standard binary search logic on B to identify $\ell = \min\{m : B[m] = 0\}$. The correctness of this search depends on the correctness of the logic SplitCheck uses to test whether a given position is 1 or 0. This correctness follows from how the algorithm assigns i and j to channels when testing a given level m. The formula used—$\lceil id/2^{\lg \mathcal{C} - m} \rceil$—assigns i and j to the same channel iff their paths share a node at that level of $T_{\mathcal{C}}$ (that is, if $B[m] = 1$). The nodes then use collision detection to determine whether or not they share the same channel for a given check.

Once we have established that the nodes effectively set $L = \ell$ with their call to SplitCheck, the correctness of the whole routine follows. By definition, i and j share a parent at level $\ell - 1$ in $T_{\mathcal{C}}$ and therefore only one these two nodes is the common parent's left child—the node that will go on to transmit alone on channel 1 and solve the problem.

Finally, we note that the time required to complete this search is in $O(\log h) = O(\log \log \mathcal{C}) = O(\log \log n)$, where the final step follows from our assumption that $\mathcal{C} \leq n$. □

5. CONTENTION RESOLUTION FOR ANY NUMBER OF NODES

We now present our main algorithm which solves contention resolution for any number of active nodes. The algorithm consists of three steps that are executed one after another in a synchronized manner. The first step leverages a straightforward *knock out* strategy to reduce the number of active nodes to a count in $O(\log n)$. (This preliminary reduction will simplify the steps that follow.) The second step reassigns active nodes unique ids from the space $[\mathcal{C}/2]$ (further reducing the number of active nodes if needed to enable this task). The third step leverages the guarantees of the two that precede it to solve the contention resolution problem.

```
Reduce (for active node $v$)
$\hat{n} \leftarrow n, r \leftarrow 1$
do
  repeat twice:
    $v$ broadcasts on channel 1 with probability $1/\hat{n}$
    if $v$ broadcasts without collision
      $v$ become leader and terminates
    else if $v$ receives and does not hear silence
      $v$ becomes inactive and terminates
  $r \leftarrow r + 1$
  $\hat{n} \leftarrow \sqrt{\hat{n}}$
while $r \leq \lceil \lg \lg n \rceil$
```

Figure 2: The Reduce Algorithm

Our main theorem follows directly from Theorems 5, 6, and 17:

Theorem 4. *In a network with up to $n \geq 1$ possible active nodes and $\mathcal{C} \geq 1$ channels, our algorithm solves contention resolution in $O\left(\frac{\log n}{\log \mathcal{C}} + (\log\log n)(\log\log\log n)\right)$ rounds, with high probability.*

5.1 Step #1: Reduce to $O(\log n)$ Active Nodes

The first step of our leader election algorithm is to reduce the number of active nodes to $O(\log n)$. This reduction will prove helpful to both of the steps that follow. We note that with collision detection, reducing a set of no more than n nodes to less than $O(\log n)$ nodes does not require multiple channels. Accordingly, the algorithm we deploy for this purpose only uses channel 1. Our algorithm Reduce uses a standard knock out strategy and reduces the active node count to $O(\log n)$ in $O(\log\log n)$ rounds. We formalize this result with the following theorem which can be shown using standard techniques from the single channel setting:

Theorem 5. *There exists a constant $\alpha \geq 1$, such that for any constant $\beta \geq 1$, when Algorithm* Reduce *terminates the number of active nodes is between 1 and $\alpha\beta \log n$, with probability at least $1 - n^{-\beta}$.*

5.2 Step #2: Reduce the Unique ID Space from $[n]$ to $[\mathcal{C}/2]$

Our goal in this second step is to continue to reduce the set of active nodes as needed until we succeed in reassigning nodes unique IDs from $[\mathcal{C}/2]$. We accomplish this task with an algorithm we call IDReduction, which guarantees to complete the needed reduction/renaming in $O(\log n/\log \mathcal{C})$ rounds, with high probability. We describe and analyze the algorithm below.

The IDReduction Algorithm. This algorithm alternates between *renaming* and *reduction* phases. The renaming phase is implemented by a pair of rounds. During the first round of the pair, all nodes that are still active at the beginning of the renaming phase choose a channel from the range $[\mathcal{C}/2]$ with uniform randomness and transmit. If a node detects it is alone on some channel i, it adopts i as its unique id. In the next round of the pair all nodes go to channel 1. Any node that adopted a unique id in the preceding round transmits. If there are any transmissions, the algorithm is over, and only those nodes who transmitted remain active (with their recently adopted ids as their new unique id).

The reduction phase requires only a single round. During a reduction phase round, all nodes that are still active transmit with probability $1/k$ on channel 1, where $k = \sqrt{\mathcal{C}}/144$. Nodes that do not transmit receive on channel 1. If there is at least one transmission, then any active node that did not transmit becomes inactive. Otherwise, the set of active nodes does not change in this round.

Analysis. In the following, we use the notation A_r to represent the set of active nodes in the beginning of the r^{th} round of IDReduction. Note that $\ldots \subseteq A_{r+1} \subseteq A_r \subseteq \ldots \subseteq A_1 = A$. In the following, we assume $\mathcal{C}$ is a sufficiently large such that $k = \sqrt{\mathcal{C}}/144 \geq 3$. (Spread throughout this analysis are several other places where we similarly assume $\mathcal{C}$ is larger than some fixed constant.) We note that we are always safe to fix a constant lower bound for $\mathcal{C}$, as when $\mathcal{C} = O(1)$, the lower bound simplifies to $\Omega(\log n)$, which we can match with the well-known $O(\log n)$ contention resolution algorithm that is optimal for the single channel case. Finally, to simplify notation, the theorem, lemma, and corollary statements that follow, when we claim a result holds *with high probability*, we mean that it holds with probability at least $1 - n^{-\beta}$, where $\beta \geq 1$ is a constant that increases along with the constants hidden within the asymptotic time complexity.

We begin our analysis by stating the main theorem. Its correctness follows directly from Corollary 8 and Lemma 10 which we describe and prove below.

Theorem 6. *Assume $|A| = O(\log n)$. With high probability:* IDReduction *terminates in $O(\log n/\log \mathcal{C})$ rounds with no more than $\mathcal{C}/2$ active nodes, each with a unique ID from $[\mathcal{C}/2]$.*

Our analysis proceeds in two pieces. In more detail, first we show that the reduction rounds reduce the number of active nodes below $\mathcal{C}/6$ within $O(\log n/\log \mathcal{C})$ rounds. Second, we show that once we have less than $\mathcal{C}/6$ participants, the renaming rounds will succeed within an additional $O(\log n/\log \mathcal{C})$ rounds. The below lemma addresses the first piece. The main idea is to note that when the number of active nodes is large compared to a target of $\approx k \log k$ (where k comes from the knock probability $1/k$ used in the reduction rounds in IDReduction) the reduction rounds achieve big reductions with high probability after only a small number of rounds. As the active node count approaches the target, the number of rounds required for high probability increases. We show that this count falls fast enough that the total number of rounds spent reducing does not sum to something too large.

Lemma 7. *Assume $|A| = O(\log n)$. There exists a round $\hat{r} = O(\log n/\log k)$, such that with high probability: $|A_{\hat{r}}| \leq 24k \log k$.*

Proof. Let $c_1 = 24$ be the constant from the lemma statement; i.e., our goal can be stated as achieving $|A_{\hat{r}}| \leq c_1 k \log k$. By assumption, $|A| \leq c' \log n$ for some constant $c' \geq 0$. Therefore, it clearly follows that $|A| \leq \gamma \log n$ for $\gamma = \max\{24, c'\}$. We also assume $k < n$. If this was not true, then the assumption that $|A| = O(\log n)$ would imply that $|A| \leq c' \log k$ which is smaller than $24k \log k$ for any constant

c' once $\mathcal{C}$ is a sufficiently large constant, which would make the lemma trivially true. We also assume $\mathcal{C}$ is a sufficiently large constant that $k \geq 3$. An immediate corollary from this assumption is that $|A| \leq \gamma k \log n$. Because the set of active nodes can never increase as the execution continues, this upper bound always holds.

To begin our proof argument, we note that in each round r for which we have not yet sufficiently reduced the number of active nodes, we know: $c_1 k \log k < |A_r| \leq \gamma k \log n$. For each such round r, therefore, we can express $|A_r|$ as the quantity $\gamma k(\log n/q(r))$, for some value $q(r)$ that is from the range 1 to $T = (\gamma \log n)/(c_1 \log k)$. (We know $T > 1$ because $\gamma \geq c_1$ and $n > k$.) Furthermore, we know the sequence $q(1), q(2), q(3), ...,$ is strictly non-decreasing, as the active node count can never grow.

Fix some reduction round r such that we are not yet done. Let X_r be the number of nodes that choose to transmit during this round. We know: $E(X_r) = |A_r|/k = \gamma(\log n/q(r))$.

Because the transmission decisions are independent, we can express X_r as the sum of independent indicator variables describing the nodes' transmission decisions, and can therefore apply a Chernoff bound to achieve concentration. We use the following special form:

$$\Pr[|X_r - E(X_r)| \geq E(X_r)/2] < 2e^{-E(X_r)/12}.$$

Notice, if $|X_r - E(X_r)| < E(X_r)/2$, then because $k \geq 3$ and $E(X_r) \geq 2$, it holds that we have reduced the set of active nodes by at least a factor of 2; i.e., $|A_r|/|A_{r+1}| \geq 2$. (The former assumption ensures $(3/2)E(X_r)$ is not too large, and the latter ensures that $(1/2)E(X_r) \geq 1$.)

We want to bound the probability that $|A_r|/|A_{r+x}| < 2$, for some $x \geq 1$. That is, we want to bound the probability that after x rounds we still have not yet reduced the active node count at the start of the interval by at least a factor of 2. To bound this probability, we instead consider the *harder* model where if a round is not good, then no node becomes inactive. In this harder model, the probability that we have x reduction rounds in a row that are *not good* is less than:

$$\begin{aligned}\left(2\exp\left\{-\frac{E(X_r)}{12}\right\}\right)^x &< \left(\exp\left\{-\tfrac{E(X_r)}{24}\right\}\right)^x \\ &= \exp\left\{-\tfrac{x\gamma \log e \ln n}{24 q(r)}\right\},\end{aligned}$$

where the first inequality loosely holds so long as we assume that $E(X_r) \geq 24$ (for all values of $q(r)$ from 1 to T the expectation is greater than this value). Next notice for that for $x \geq \frac{q(r)c_2 24}{\gamma \log e}$, defined for any constant $c_2 \geq 1$, this probability is less than n^{-c_2}. Put another way, starting from any given round r, with high probability, after $\Theta(q(r))$ reduction rounds we have reduced the active node count by a factor of at least 2. (And as we increase the constant c_2 hidden in the $\Theta(q(r))$ term, the exponent on the failure probability increases as well.)

We next partition reduction rounds into intervals, where the first interval starts in the first reduction round, and the start of interval $i+1$, for $i \geq 1$, corresponds to the first round after which the number of active nodes is at least a factor of 2 smaller than at the start of interval i. Let r_i be the reduction round at the start of interval i. Let $\hat{\imath}$ be the latest interval before we fall below our target threshold of $c_1 k \log k$.

Consider the sequence of $q(r)$ values associated with the size of the active node sets during these intervals' start rounds: $q(r_1), q(r_2), ..., q(r_{\hat{\imath}})$. We note two things about this sequence. First, by our definition of an interval, for each i, $q(r_i) \leq q(r_{i+1})/2$. That is, the values at least double between intervals, corresponding to the node set size reducing by at least a factor of 2. We also note that by definition: $q(r_{\hat{\imath}}) = O(T) = O(\log n/\log k)$.

Leveraging our above probabilistic analysis, we note that for each interval i, with high probability, it takes only $O(q(r_i))$ rounds to get to interval $i+1$. Therefore, by a union bound, this is true of all intervals, also with high probability. Under this assumption, we can upper bound the number of rounds required to get through all $O(T)$ intervals with the summation $c + 2c + 4c + ... + T$, for some constant $c \geq 1$. This time complexity simplifies to $O(T) = O(\log n/\log k)$—as needed. $\square$

Our goal is to get the number of active nodes below $\mathcal{C}/6$. Notice, however, that for our definition $k = \sqrt{\mathcal{C}}/144$, the $\leq 24k\log k$ nodes guaranteed by Lemma 7 reduces to no more than:

$$\begin{aligned}24k\log k &= 24(\sqrt{\mathcal{C}}/144)\log(\sqrt{\mathcal{C}}/144) < 24(\sqrt{\mathcal{C}}/144)\log(\sqrt{\mathcal{C}}) \\ &\leq (1/6)\sqrt{\mathcal{C}}\log(\sqrt{\mathcal{C}}) < \mathcal{C}/6.\end{aligned}$$

The following corollary is a direct implication of Lemma 7 and the above inequality and the fact $O\left(\frac{\log n}{\log k}\right) = O\left(\frac{\log n}{\log \mathcal{C}}\right)$:

COROLLARY 8. *Assume $|A| = O(\log n)$. There exists a round $\hat{r} = O\left(\frac{\log n}{\log \mathcal{C}}\right)$, such that with high probability: $|A_{\hat{r}}| < \mathcal{C}/6$.*

We are left now to show that if the number of active nodes reduces below $\mathcal{C}/6$, renaming becomes likely to succeed. Before making our main argument to this effect, we first isolate a useful balls-in-bins claim that the argument will leverage.

LEMMA 9. *Assume you throw b balls into m bins such that $b = m/\beta$, where $3 \leq \beta < m$. The probability that* no *ball ends up alone in a bin is less than $\frac{1}{2^{b/2}}$.*

PROOF. To achieve our bound we will bound the probability that $(m - m/(2\beta))$ of m bins are empty. Notice, this event must hold if no ball ends up alone, as if more than $m/(2\beta)$ bins are full, then by a straightforward counting argument, at least one ball must end up alone.

Continuing with our calculation, consider a fixed set of $(m - m/(2\beta))$ empty bins. For each ball, the probability that the ball misses these empty bins and falls into one of the $m/(2\beta)$ *free* bins it is allowed to occupy, is $1/(2\beta)$. The probability that *all* balls land in a free bin is therefore $1/(2\beta)^{m/\beta}$. This probability concerns only a single set of empty bins. We now consider all such sets. In more detail, there are $\binom{m}{m-m/(2\beta)}$ different ways to select our $(m - m/(2\beta))$ empty bins. Taking a union bound over these different selections, we derive that the probability that there is at least one selection of $(m - m/(2\beta))$ empty bins that remains empty after throwing m/β balls, is upper bounded as follows:

$$\frac{\binom{m}{m-\frac{m}{2\beta}}}{(2\beta)^{\frac{m}{\beta}}} = \frac{\binom{m}{\frac{m}{2\beta}}}{(2\beta)^{\frac{m}{\beta}}} \leq \frac{(2\beta e)^{\frac{m}{2\beta}}}{(2\beta)^{\frac{m}{\beta}}} = \left(\frac{e}{2\beta}\right)^{\frac{m}{2\beta}} < \frac{1}{2^{\frac{m}{2\beta}}},$$

where the last step holds because we assume $\beta \geq 3 > e \Rightarrow e/(2\beta) < (1/2)$. To achieve the bound stated in the lemma our final step is to substitute the definition $b = m/\beta$. $\square$

We now leverage Lemma 9 to argue that renaming will succeed once the number of active nodes is small enough for the above balls-in-bins argument to give us a sufficiently high probability of success. The probability we seek is something at least $(1 - (1/\mathcal{C}))$. If we succeed with this probability, then the probability we fail $O(\log n/\log \mathcal{C})$ times in a row becomes small in n,

Lemma 10. *Fix a round r such that $|A_r| \leq \mathcal{C}/6$. With high probability, renaming succeeds and the algorithm terminates within $O(\log n/\log \mathcal{C})$ rounds.*

Proof. In renaming rounds, active nodes each select a channel with uniform randomness. If any node is alone on its chosen channel the algorithm terminates. We will prove here that once we fall below $\mathcal{C}/6$ nodes, this will occur with high probability after the stated number of attempts. To do so, fix some renaming round t. Let $n' = |A_t|$ be the number of nodes still active during this round. Assume $1 < n' \leq \mathcal{C}/6$ (the lower bound is safe to assume as if $n' = 1$ it is trivial to show we will terminate in this round). Let $\hat{\mathcal{C}} = \mathcal{C}/2$. We consider two cases concerning the size of n', and will show for each that the probability we do *not* terminate in t is less than $1/2^{\log \hat{\mathcal{C}}/2}$. We will then show this probability is sufficiently low to terminate within the time claimed by the lemma statement with high probability.

The first case for the size of n' is when $n' \leq \sqrt{\hat{\mathcal{C}}}$. Fix any node $i \in A_t$. Let x be the channel it selects in this round. The probability that some other $j \in A_t$ also chooses x is $1/\hat{\mathcal{C}}$. By a union bound, the probability that at least one node in A_t chooses x is less than: $n'/\hat{\mathcal{C}} \leq 1/\sqrt{\hat{\mathcal{C}}} = 1/2^{\lg(\sqrt{\hat{\mathcal{C}}})} = 1/2^{\lg \hat{\mathcal{C}}/2}$, as needed.

The second case is when $n' > \sqrt{\hat{\mathcal{C}}}$. Here we can apply Lemma 9 to the problem of throwing n' balls in $\hat{\mathcal{C}}$ bins (which requires our assumption that $1 < n' \leq \mathcal{C}/6$ which implies $n' \leq \hat{\mathcal{C}}/3$). This lemma tells us that the probability that *no* ball is alone is less than $\frac{1}{2^{n'/2}}$. Given our assumption that $n' > \sqrt{\hat{\mathcal{C}}}$, it follows that $1/2^{n'/2} < 1/2^{\sqrt{\hat{\mathcal{C}}}/2} < 1/2^{\log \hat{\mathcal{C}}/2}$. This final step requires the assumption that $\log \hat{\mathcal{C}} \leq \sqrt{\hat{\mathcal{C}}}$, which is always true for $\hat{\mathcal{C}} \geq 16$. Given that we assumed $\mathcal{C} \geq 32$ at the beginning of this section, it follows that $\hat{\mathcal{C}} \geq 16$.

We have just shown that in every round (once the number of active nodes is sufficiently small), we fail to terminate with probability less than $1/2^{\log \hat{\mathcal{C}}/2}$. The probability that we fail to terminate for $T = (c \log n)/\log \hat{\mathcal{C}}$ renaming rounds in a row (for a fixed constant $c \geq 1$), therefore, is less than:

$$\left(\left(\frac{1}{2}\right)^{\frac{\log \hat{\mathcal{C}}}{2}}\right)^T = \left(\left(\frac{1}{2}\right)^{\frac{\log \hat{\mathcal{C}}}{2}}\right)^{\frac{c \log n}{\log \hat{\mathcal{C}}}} = \left(\frac{1}{2}\right)^{\frac{c \log n}{2}} = \left(\frac{1}{2}\right)^{\frac{c}{2}}.$$

Because $T = O(\log n/\log \mathcal{C})$, it follows that the probability we fail to terminate in $T = O(\log n/\log \mathcal{C})$ rounds is polynomially small in n (with an exponent that increases with the constant c in T). $\square$

5.3 Step #3: Elect a Leader

The result of the first and second step of our algorithm is that we now have $x \leq \mathcal{C}/2$ active nodes with unique IDs in $[\mathcal{C}/2]$, and in addition $x = O(\log n)$ with high probability. These nodes participate in the third and final step of our algorithm, which we call LeafElection. This algorithm *deterministically* elects a leader using a tree of channels (i.e., a tree with $\leq \mathcal{C}$ nodes for which we have assigned a unique channel for each tree node). This algorithm runs in $O(\log h \log\log x)$ rounds, where x is the starting number of active nodes and $h = \lg \mathcal{C}$ is the height of the tree of channels. As before, we shall assume without loss of generality that $\mathcal{C}$ is a power of 2.

Algorithm Description. Before describing the algorithm mechanics (see Figure 3 for pseudocode), we first discuss the tree of channels (equiv., channel tree) the algorithm uses. As in Section 4 (the two node algorithm), we map channels to a complete binary tree T with $\mathcal{C}/2$ leaves, with channels identified in the same canonical fashion as before. In certain rounds, it will also be convenient to choose a single channel to represent each level (or row) in the tree; to do so, we choose the leftmost tree node at that level to act as the level's representative channel.

The core behavior of our algorithm is to *coalesce* active nodes into larger and larger groups we call *cohorts* such that every node in the same cohort has a distinct ID from a known ID space the same size as the cohort. Eventually, only one cohort remains, and the distinct IDs can be used to identify the leader for the whole network. The main point of the cohorts is to accelerate the binary searches used to identify key levels in the channel tree.

More precisely, the main algorithm LeafElection consists of a sequence of iterations or *phases*. Initially, all active nodes belong to their own cohorts of size 1. We associate with each cohort a distinct tree node *cNode*, which is the least common ancestor of all active nodes in the cohort. Initially, therefore, the cohorts are associated with the leaves of the channel tree. The algorithm maintains the invariant that at the start of the ith phase all active cohorts have exactly 2^{i-1} active nodes, and each active node in a particular cohort has a distinct *cID* (or cohort ID) from $[2^{i-1}]$. We call the node with *cID* $= 1$ in the cohort the *cohort master*.

Each phase begins by testing whether more than one cohort exists by having the cohort masters broadcast on the root channel. If there is more than one cohort, then the phase must guarantee that (1) at least one cohort exists at the end of the phase, and (2) all cohorts become twice as large. We achieve this goal by pairing together some cohorts and having others become inactive. In particular, we employ SplitSearch to identify the level ℓ closest to the root such that all cohorts have a different level-ℓ ancestor. Since we choose the smallest such level, there exists at least one pair (and possibly many pairs) of cohorts that have the common level-$(\ell - 1)$ ancestor. To identify these pairs, each cohort master broadcasts on the channel for its level-$(\ell - 1)$ ancestor (and all other members of the cohort listen). If there is a collision, then this cohort can be paired; otherwise, it becomes inactive. To pair the cohorts, we observe that one cohort is in the ancestor's left subtree, whereas the other must be in the right subtree, so we increase the *cIDs* of nodes in the right subtree by the cohort size. Finally, the cohort tree node *cNode* can be updated to this ancestor.

```
LeafElection (for active node $v$)
$cSize \leftarrow 1$             // size of all active cohorts
$cNode \leftarrow v$'s leaf   // the subtree for $v$'s cohort
$cID \leftarrow 1$              // $v$'s distinct id within its cohort
repeat
  if $cID = 1$
    $v$ broadcasts on root's channel
  else $v$ listens on root's channel
  if there was no collision
    the lone broadcaster is the leader
  else let $\ell_{\max} = cNodes$'s level
    $\ell \leftarrow$ SplitSearch$(0, \ell_{\max}, cSize, cNode, cID)$
    if $cID = 1$
      $v$ broadcasts on $a_{\ell-1}(v)$'s channel
    else $v$ listens on $a_{\ell-1}(v)$'s channel
    if collision at $a_{\ell-1}(v)$
      if $v$ in right subtree of $a_{\ell-1}(v)$
        $cID \leftarrow cID + cSize$
      $cSize \leftarrow 2cSize$
      $cNode \leftarrow a_{\ell-1}(v)$
    else $v$ becomes inactive
until leader declared

SplitSearch$(\ell_{\min}, \ell_{\max}, cSize, cNode, cID)$ for active $v$
// return level closest to root where all subtrees
  have $\leq 1$ cohort

if $\ell_{\max} = \ell_{\min} + 1$ return $\ell_{\max}$
else let $probedist = \lceil(\ell_{\max} - \ell_{\min})/cSize\rceil$
  let $k$ be smallest value such that
  $\ell_{\min} + k \cdot probedist \geq \ell_{\max}$
  for $i < k$, define $\ell_i$ as $\ell_i = \ell_{\min} + i \cdot probedist$
  define $\ell_k = \ell_{\max}$
  if $cID < k$
    CheckLevel$(\ell_{cID})$
    CheckLevel$(\ell_{cID+1})$
  else do nothing for 4 rounds
  if $cID = 1$ and the first check returned "no collision"
    announce 0 on channel $cNode$ and set $i \leftarrow 0$
  else if $cID < k$ and only the first check returned
  "collision"
    announce $cID$ on channel $cNode$ and set $i \leftarrow cID$
  else listen to $cNode$ and set $i$ to the announced value
  return SplitSearch$(\ell_i, \ell_{i+1}, cSize, cNode, cID)$

CheckLevel$(\ell)$ for active node $v$

broadcast on $a_\ell(v)$
if a collision occurred on $a_\ell(v)$
  broadcast on the channel for row $\ell$
else listen on the channel for row $\ell$
if the channel for row $\ell$ was silent
  return "no collision"
else return "collision"
```

Figure 3: The Algorithm LeafElection. The notation $a_\ell(v)$ refers to v's level-ℓ (i.e., depth-ℓ) ancestor in the tree of channels.

The SplitSearch routine is similar to SplitCheck in Section 4, with the exception of two key differences. First, the paths down to active cohorts may diverge at different points, but we wish to identify the smallest level ℓ globally where all have diverged. We thus employ a slightly more complicated test, called CheckLevel to test whether any nodes share an ancestor at level ℓ. The test consists of two rounds. During the first round, one node per cohort broadcasts on its level-ℓ ancestor. Some nodes may observe a collision, whereas others may not. To arrive at the same conclusion, any collisions are announced on the row channel for level-ℓ.

The second key difference in SplitSearch is that we exploit the large size of cohorts to accelerate the search. In particular, let p be the size of cohorts. Then the search is a $(p+1)$-ary search, instead of a binary search, adapted from Snir's [16] CREW parallel search. The search takes as input a range of levels $(\ell_{\min}, \ell_{\max}]$ to search. This range is then subdivided into $p+1$ subranges of roughly the same size, given by $(\ell_0 = \ell_{\min}, \ell_1], (\ell_1, \ell_2], (\ell_2, \ell_3], \ldots, (\ell_{k-1}, \ell_k = \ell_{\max}]$. (Usually $k = p+1$, except within one recursion of the base case where k can be smaller.) The observation is that each subrange can be tested independently—if there is a collision at level ℓ_i but no collision at level ℓ_{i+1}, then $(\ell_i, \ell_{i+1}]$ is the subrange to search. Thus, we can test the subranges in parallel by assigning one node per cohort to each range according to *cID*s. Once the correct subrange has been identified by one node per cohort, that node announces the range to its comrades on the cohort tree node *cNode*. Note that this step requires listeners to read and interpret the message, not just listen for collisions. In this way, all nodes know which subrange to recurse in, and the size of the range has been reduced by roughly a $p+1$ factor.

Analysis. To prove correctness, we will argue that the following structural properties are invariant across phases. Assuming the properties hold for each iteration, we first prove that subroutines operate correctly (Lemmas 12 and 13). We then inductively argue that Property 11 indeed holds with Lemma 14.

Property 11. *Let i be the phase number of* LeafElection*:*

- *All active nodes belong to a cohort.*
- *Each cohort has exactly cSize $= 2^{i-1}$ active nodes as members.*
- *Each member of a particular cohort has a distinct identifier cID $\in [2^{i-1}]$.*
- *The cohort node is the treenode that is the least common ancestor of all members (which correspond to leaves). All cohort nodes occur at distinct nodes in the same level of the tree. Moreover, all members of a cohort correctly identify their cohort node with cNode.*

Lemma 12. *Suppose that Property 11 holds and that exactly one node in each active cohort performs* CheckLevel(ℓ)*. Then the return value is "no collision" if and only if all cohorts have distinct level-ℓ ancestors.*

Proof. By assumption, all cohort nodes occur at the same level. If ℓ is a descendent level (i.e., ℓ is at least the level of cohort nodes), then CheckLevel correctly returns "no collision"—by Property 11, each participating node is a descendent of a distinct cohort node, so they broadcast on different channels.

Suppose instead that ℓ is smaller than the cohort nodes' level. If two cohorts share an ancestor at level ℓ, then by Property 11 all members of those cohorts also share that ancestor. Thus, the first broadcast observes a collision, which is advertised to all other cohorts in the second broadcast. If no cohorts share an ancestor at that level, then no collision is observed by anyone. □

LEMMA 13. *Suppose that Property 11 holds when* SPLITSEARCH *is called from* LEAFELECTION. *Then* SPLITSEARCH *correctly returns the smallest level ℓ (nearest to root) such that all cohorts have distinct level-ℓ ancestors.*

PROOF. The proof is by induction over the recursive calls. The hypothesis is that on each call 1) all concurrent calls are synchronized and use the same values of $\ell_{\min}$ and $\ell_{\max}$, 2) $\ell_{\min} < \ell$, and 3) $\ell \leq \ell_{\max}$. Assuming the hypothesis, the search only terminates if the correct value is identified. It is easy to verify that the search eventually terminates because the range gets strictly smaller on each recursive call.

For the base case, (1) follows from the fact that all cohort nodes have the same level, and hence all active nodes agree on $\ell_{\max}$. (3) is true trivially since all cohort nodes are distinct nodes at level $\ell_{\max}$. Since the search only executes if there was a collision at the root, we know $0 < \ell$ and hence (2) holds.

For the inductive step, it is easy to verify that $\ell_{\min} = \ell_0 < \ell_1 < \cdots \ell_k = \ell_{\max}$. Moreover, due to distinct *cID*s (Property 11), CHECKLEVEL(ℓ_i) is performed by exactly one node in each cohort at a time. Thus, by Lemma 12 these calls perform the correct answers, with the corresponding nodes in each cohort observing the same answers. Moreover, CHECKLEVEL(ℓ_i) returns "collision" for all $\ell_i < \ell$ and "no collision" for $\ell_i \geq \ell$, so only one subrange can be identified for the recursive search and hence only one node per cohort makes an announcement. (This is essentially the same argument as Section 4 as well as the parallel search [16].) Finally, we must verify that the range is announced without collision and that all nodes in the cohort listen, which follows from the assumption that each cohort has a different *cNode*. □

LEMMA 14. *Property 11 holds at the start of the ith phase of* LEAFELECTION.

PROOF. The proof is by induction. By assumption that each active node is a separate leaf of the tree, the property holds trivially initially with all cohorts consisting of a single active node.

Suppose the property holds at the start of the ith phase. Then we must show that it holds at the start of the next iteration. Since SPLITSEARCH returns the correct answer (Lemma 13), we know that (1) all cohorts have distinct level-ℓ ancestors, (2) at least one level-$(\ell - 1)$ treenode has multiple descendent cohorts, and (3) all such level-$(\ell - 1)$ treenodes have exactly two descendent cohorts, one in the left and one in the right subtree. Since exactly one node per cohort broadcasts on its level-$(\ell - 1)$ ancestor, the cohorts observe a collision if and only and only if they share a level-$(\ell - 1)$ ancestor with another cohort. Thus, cohorts remain active if and only if they can be paired with another cohort, thereby doubling the size of the cohort. Since the *cID*s for a cohort are by inductive assumption distinct values from [*cSize*], adding *cSize* to the IDs of one of the paired cohorts preserves distinctness. Moreover, the new *cNode* is indeed the least common ancestor of both cohorts, and hence of all nodes therein. □

We are now ready to bound the performance of LEAFELECTION. We first bound the number of phases as a direct corollary of Property 11. Then we bound the cost of the searches.

COROLLARY 15. *If* LEAFELECTION *begins with x active nodes assigned to distinct leaves of the channel tree, then it correctly identifies a leader in $O(\log x)$ phases.*

PROOF. By Lemma 14, Property 11 holds, and hence at the start of the ith phase all cohorts have size 2^{i-1}. Thus, there cannot be more than $\lg x + 1$ phases. □

LEMMA 16. *During the ith phase,* SPLITSEARCH *completes in $O((1/i)\log h)$ rounds, where $h = \lg \mathcal{C}$ is the height of the channel tree.*

PROOF. With each recursive call, the size of the level subrange to search becomes $\lceil(\ell_{\max} - \ell_{\min})/\mathit{cSize}\rceil$, i.e., reducing by at least a $\Theta(\mathit{cSize}) = \Theta(2^i) + 1$ factor. Thus, the number of recursive calls is $O(\log_{2^i+1} h) = O(\log h/\log 2^i) = O((1/i)\log h)$. Noting that each call is a constant number (specifically 5) of rounds completes the proof. □

Finally, we have our main theorem for this step of leader election:

THEOREM 17. *If* LEAFELECTION *begins with x active nodes assigned to distinct leaves of the channel tree, then it correctly identifies a leader in $O(\log h \log\log x)$ phases, where $h = \lg \mathcal{C}$ is the height of the channel tree.*

PROOF. Correctness follows from the Lemma 14. To prove the performance, we observe that the work of each of the $\lg x$ phases is dominated by the SPLITSEARCH. Applying Corollary 15 and Lemma 16, we conclude that the total number of rounds is $\sum_{i=1}^{O(\log x)} O((1/i)\log h) = O\left(\log h \sum_{i=1}^{O(\log x)}(1/i)\right) = O(\log h \log\log x)$. □

Recall that if the previous steps are successful, then $x = O(\log n)$, and hence this bound reduces to:

$$O(\log\log \mathcal{C} \cdot \log\log\log n) = O(\log\log n \cdot \log\log\log n).$$

6. CONCLUSION

In this paper, we study the classical *contention resolution* problem, in which an unknown subset of n possible nodes are activated and connected on shared channels with the goal of breaking symmetry. We considered, in particular, the case where nodes have access to multiple channels and collision detectors. We described and analyzed new algorithms that match the relevant lower bound for the restricted case of two active nodes, and for the general case come within a factor of $\log\log n$. These results help advance one of the few remaining major open cases for the longstanding study of contention resolution.

The obvious next step is to tackle this final small gap between lower and upper bounds. We predict that the lower bound is in fact tight. Proving this (somewhat tentative) assertion, however, will require more advanced algorithmic techniques. Another open problem is to tackle contention resolution in this setting with respect to expected time of termination. Not much is known about expected time solutions in this case. One reason for this omission is that even *without* collision detection, the best expected time solutions are really fast, reaching $O(1)$ expected complexity with as few as $\log n$ channels. This leaves only a small band of parameters for which the addition of collision detection might possibly improve performance.

7. ACKNOWLEGEMENTS

The research in this paper has been supported by NSF grants CCF 1320279 and CCF 1314633.

8. REFERENCES

[1] N. Abramson. The ALOHA System: Another Alternative for Computer Communications. In *Proceedings of the Fall Joint Computer Conference*, 1970.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the Time Complexity of Broadcast in Radio Networks: an Exponential Gap Between Determinism and Randomization. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 1987.

[3] S. Daum, S. Gilbert, F. Kuhn, and C. Newport. Leader Election in Shared Spectrum Networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2012.

[4] S. Daum, F. Kuhn, and C. Newport. Efficient Symmetry Breaking in Multi-Channel Radio Networks. In *Proceedings of the International Symposium on Distributed Computing*, 2012.

[5] W. D.E. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal on Computing*, 15(2):468–477, 1986.

[6] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Lower Bounds for Clear Transmissions in Radio Networks. In *Proceedings of the Latin American Symposium on Theoretical Informatics*, 2006.

[7] R. Gallager. A Perspective on Multiaccess Channels. *IEEE Transactions on Information Theory*, 31(2):124–142, 1985.

[8] L. Gasieniec, A. Pelc, and D. Peleg. The Wakeup Problem in Synchronous Broadcast Systems. *SIAM Journal on Discrete Mathematics*, 14(2):207–222, 2001.

[9] A. Greenberg and S. Winograd. A Lower Bound on the Time Needed in the Worst Case to Resolve Conflicts Deterministically in Multiple Access Channels. *Journal of the ACM*, 32(3):589–596, 1985.

[10] B. Hajek and T. van Loon. Decentralized Dynamic Control of a Multiaccess Broadcast Channel. *IEEE Transactions on Automatic Control*, 27(3):559–569, 1982.

[11] T. Jurdziński and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Algorithms and Computation*, pages 535–549. Springer, 2002.

[12] M. Kaplan. A Sufficient Condition for Non-Ergodicity of a Markov Chain. *IEEE Transactions on Information Theory*, IT-25:470–471, July 1979.

[13] J. Komlos and A. Greenberg. An Asymptotically Nonadaptive Algorithm for Conflict Resolution in Multiple-Access Channels. *IEEE Transactions on Information Theory*, 31(2):302–306, 1985.

[14] C. Newport. Radio Network Lower Bounds Made Easy. In *Proceedings of the International Symposium on Distributed Computing*, 2014.

[15] L. G. Roberts. ALOHA Packet System with and Without Slots and Capture. *ACM SIGCOMM Computer Communication Review*, 5(2):28–42, 1975.

[16] M. Snir. On Parallel Searching. *SIAM Journal on Computing*, 14(3):688–708, 1985.