

Optimal leader election in multi-hop radio networks ^{*†}

Artur Czumaj Peter Davies

Department of Computer Science
Centre for Discrete Mathematics and its Applications
University of Warwick

Abstract

We present two optimal randomized leader election algorithms for multi-hop radio networks, which run in expected time asymptotically equal to the time required to broadcast one message to the entire network. We first observe that, under certain assumptions, a simulation approach of Bar-Yehuda, Golreich and Itai (1991) can be used to obtain an algorithm that for directed and undirected networks elects a leader in $O(D \log \frac{n}{D} + \log^2 n)$ expected time, where n is the number of the nodes and D is the eccentricity or the diameter of the network. We then extend this approach and present a second algorithm, which operates on undirected multi-hop radio networks with collision detection and elects a leader in $O(D + \log n)$ expected run-time. This algorithm in fact operates on the beep model, a strictly weaker model in which nodes can only communicate via beeps or silence. Both of these algorithms are optimal; no optimal expected-time algorithms for these models have been previously known.

We further apply our techniques to design an algorithm that is quicker to achieve leader election with high probability. We give an algorithm for the model without collision detection which always runs in time $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$, and succeeds with high probability. While non-optimal, and indeed slightly slower than the algorithm of Ghaffari and Haeupler (2013), it has the advantage of working in directed networks; it is the fastest known leader election algorithm to achieve a high-probability bound in such circumstances.

^{*}Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP).

[†]Contact information: {A.Czumaj, P.W.Davies}@warwick.ac.uk. Phone: +44 24 7657 3796.

1 Introduction

Leader election is the problem of ensuring that all nodes agree on a single node to be designated leader. Specifically, at the conclusion of a leader election algorithm, all nodes should output the same ID, and precisely one node should identify this ID as its own. Leader election is a fundamental primitive in distributed computations and, as the most fundamental means of breaking symmetry within radio networks, it is used as a preliminary step in many more complex communication tasks. For example, many fast multi-message communication protocols require construction of a breadth-first search tree (or some similar variant), which in turn requires a single node to act as source (for more examples, cf. [4, 9, 10], and the references therein).

In this paper we present a simple framework combining the simulation approach of Bar-Yehuda et al. [2] with basic communication primitives that enables us to obtain new, optimal or almost optimal leader election algorithms.

We begin with noting that, under certain assumptions, the simulation approach of Bar-Yehuda et al. [2] can be used to obtain an optimal algorithm for the task of randomized leader election in multi-hop radio networks without collision detection (Theorem 9). The algorithm runs in $O(D \log \frac{n}{D} + \log^2 n)$ time, where n is the number of the nodes and D is the eccentricity or the diameter of the network; the complexity of this algorithm is asymptotically equal to the amount of time required to broadcast a single message. This improves by a $O(\log \log n)$ factor over the expected running time claimed by Bar-Yehuda et al. [2], and, in the case of undirected networks, by a similar $O(\log \log n)$ factor over the high probability bound of Ghaffari and Haeupler [10]. While our algorithm could be seen as a variation of the approach by Bar-Yehuda et al. [2], the obtained result has not previously been noted despite substantial research into the model.

We then present another algorithm which follows a similar outline but exploits the properties of the collision detection mechanism (Theorem 12). The algorithm operates on undirected multi-hop radio networks with collision detection (as well as the strictly weaker beep model) and has $O(D + \log n)$ expected running time, which matches the amount of time required to broadcast a single $\Theta(\log n)$ bit message to the network. It is the first known optimal leader election algorithm in this model.

For situations when a high-probability bound is required, we present an algorithm for *directed networks without collision detection* which takes a fixed $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ time and succeeds with high probability (Theorem 10). This algorithm is non-optimal, and for undirected networks slower than the algorithm of [10], but is the fastest known algorithm for high probability leader election in directed networks.

Multi-hop radio networks. We consider the classical model of *directed radio networks with unknown structure*. A *radio network* is modeled by a network (directed or undirected graph) $\mathcal{N} = (V, E)$, where the set of nodes corresponds to the set of transmitter-receiver stations. The nodes of the network are assigned different identifiers, IDs. A directed edge $(v, u) \in E$ means that the node v can send a message to the node u , whereas an undirected edge $\{v, u\} \in E$ means that the nodes v and u can exchange messages in both directions. To make the leader election problem feasible, we assume that every pair of nodes in \mathcal{N} is strongly connected or connected, depending on whether we consider directed or undirected networks.

In accordance with the standard model of unknown radio networks, we assume that a node does not have any prior knowledge about the topology of the network, its in-degree and out-degree, or the set of its neighbors. We assume that the only knowledge of each node is the size of the network n , and the *eccentricity* of the network D , which is the maximum distance between any pair of nodes in \mathcal{N} . The assumption of knowledge of n and D is integral to the result, and is common in the study of multi-hop radio networks.

We assume that all nodes have access to a global clock and work *synchronously* in discrete time steps.

When we refer to the ‘running time’ of an algorithm, we mean the number of time steps which elapse before completion (i.e. we are not concerned with the number of calculations nodes perform within time steps). In each time step a node can either *transmit* an $O(\log n)$ -bit message to all of its out-neighbors at once or can remain silent and *listen* to the messages from its in-neighbors. The distinguishing feature of radio networks is the interfering behavior of transmissions. In the most standard radio networks model, the *model without collision detection*, if a node v listens in a given round and precisely one of its in-neighbors transmits, then v receives the message. In all other cases v receives nothing; in particular, the lack of collision detection means that v is unable to distinguish between zero of its in-neighbors transmitting and more than one. The model without collision detection describes the most restrictive interfering behavior of transmissions; we also consider a less restrictive variant, the *model with collision detection*. In this model, if a node v listens in a given round then it can distinguish between zero of its in-neighbors transmitting and more than one.

Rather than directly studying the multi-hop radio network model with collision detection added, in this paper we will instead work in the strictly weaker *beep model*, introduced recently by Cornejo and Kuhn [7] as an alternative way of modeling the classical radio network model. In this model, in synchronous time steps, each node can either beep or not beep. If a node beeps, then it does not receive any information in this time step. If a node does not beep, then it can tell whether zero neighbors beeped, or whether at least one did. Any algorithm designed for the beep model can be directly used for the standard multi-hop radio network model with collision detection.

We will consider randomized algorithms for leader election and we say that a leader election algorithm *runs in expected time T* , if it finishes in expected time T , and that with high probability (w.h.p.)¹ all nodes in the network hold the same ID that identifies a single node in the network. We will say that a leader election algorithm *runs in time T with high probability (w.h.p.)* if with high probability it finishes in time T and all nodes in the network hold the same ID that identifies a single node in the network.

Previous work. The study of leader election in radio networks started in the 1970s with the *single-hop network model*, in which all nodes are directly reachable by all others (in a single hop). In this setting, in the *model with collision detection*, leader election can be performed deterministically in $O(\log n)$ time, which was proven to be optimal by Greenberg and Winograd [12]. While randomization provides no benefit if a high probability bound is required, as a $\Omega(\log n)$ lower bound also exists [11, 17], Willard [18] gave an algorithm with the expected running time of $O(\log \log n)$ and showed that this bound is also asymptotically optimal. (We note, however, that this result assume that nodes do not know the value of n , nor a linear upper bound.) For the single-hop network model *without collision detection*, deterministic leader election has complexity $\Theta(n \log n)$ [6, 15], and randomized leader election has expected time complexity $O(\log n)$ [16] and high probability time complexity $O(\log^2 n)$ [13].

While the complexity of leader election in single-hop networks is well understood, the complexity of the problem in the more general model of *multi-hop networks* has been less developed. In the seminal work initiating the study of the complexity of communication protocols in multi-hop radio networks, Bar-Yehuda et al. [2] developed a general randomized framework of simulating single-hop networks with collision detection by multi-hop networks without collision detection. The framework yields leader election algorithms for multi-hop networks (in directed and undirected networks) running in $O(T_{BC} \cdot \log \log n)$ expected time and $O(T_{BC} \cdot \log n)$ time with high probability, where T_{BC} is the time required to broadcast a message from

¹Throughout the paper we will use the phrase *with high probability* (abbreviated *w.h.p.*) to indicate the probability at least $1 - n^{-c}$, for any constant $c \geq 1$. While in our analysis, for the simplicity of presentation, we will focus only on the case $c = 1$, in all cases discussed in the paper, the high probability bound can be easily improved to be at least $1 - n^{-c}$ for any positive constant c , where larger constants c can be obtained at the expense of larger constants in the running time.

a single source to the entire network. The same authors also gave a randomized broadcasting algorithm running in $O(D \log n + \log^2 n)$ time with high probability, thereby yielding a leader election algorithm taking $O((D \log n + \log^2 n) \log \log n)$ expected time and $O(D \log^2 n + \log^3 n)$ time with high probability.

The next improvement came with faster algorithms for broadcast due to Czumaj and Rytter [8], and independently Kowalski and Pelc [14], which require only $O(D \log \frac{n}{D} + \log^2 n)$ time w.h.p. Combining these algorithms with the simulation framework of Bar-Yehuda et al., one obtains leader election algorithms (even in the model without collision detection, and in both directed and undirected graphs) running in $O((D \log \frac{n}{D} + \log^2 n) \log \log n)$ expected time and $O((D \log \frac{n}{D} + \log^2 n) \log n)$ time with high probability.

Very recently, Ghaffari and Haeupler [10] took a new approach, which yielded faster high-probability leader election algorithms in *undirected networks*. The main idea of Ghaffari and Haeupler is to randomly select a small (logarithmic) number of candidates for the leader and then repeatedly run “debates” to reduce the number of candidates to one. Standard random sampling technique allows one to choose in constant time a random set of $\Theta(\log n)$ candidates, with high probability. Then, by running a constant number of broadcasting computations and neighborhood exploration algorithms (this phase is called a “debate” in [10] and it relies heavily on the assumption that the network is undirected), one can reduce the number of candidates by a constant factor. Using this approach, Ghaffari and Haeupler [10] gave a leader election algorithm (in undirected networks) that in $O((D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log \log n, \log \frac{n}{D}\})$ rounds elects a single leader with high probability. In the model with collision detection (in fact, the *beep model* also employed in this paper), Ghaffari and Haeupler [10] used the approach above to elect a leader in $O((D + \log n \log \log n) \cdot \min\{\log \log n, \log \frac{n}{D}\})$ rounds with high probability. These algorithms are nearly optimal, and are the fastest currently known for undirected networks and for high-probability running time.

A different approach has been developed by Chlebus et al. [4], who presented a randomized leader election algorithm in undirected networks without collision detection running in the expected number of $O(n)$ rounds, or in $O(n \log n)$ rounds with high probability. This first bound is optimal in terms only of n , but does not seem tractable to parameterizations by D .

There have been also some deterministic algorithms for leader election in multi-hop radio networks. Kowalski and Pelc [15] presented a deterministic leader election algorithm in undirected networks with collision detection running in the optimal $O(n)$ time. For the model without collision detection, it is known that any deterministic leader election requires $\Omega(n \log n)$ time [6], and Chlebus et al. [4] gave a deterministic leader election algorithm in undirected networks running in $O(n \log^{1.5} n \sqrt{\log \log n})$ time. We note also that Förster et al. [9] presented a deterministic algorithm in the beep model running in $O(D \log n)$ time.

Let us finally emphasize that leader election takes at least as long as broadcasting an $\Omega(\log n)$ bit message (cf. [10, Section 3.2]), and, as such, is subject to the lower bounds $\Omega(D \log \frac{n}{D} + \log^2 n)$ for broadcasting without collision detection (cf. [1, 16]), and $\Omega(D + \log n)$ for broadcasting with collision detection or in the beep model [11, 17].

Relation of our approach to earlier works. While our main approach for randomized leader election in multi-hop radio networks without collision detection (Theorem 9) essentially follows the approach of Bar-Yehuda et al. [2], and while the result in Theorem 9 could be deduced from [2], to the best of our knowledge such implication has not been published before (and in fact this has been indirectly mentioned recently as an open problem in [10]). The main contribution of our paper is the observation that this approach can be applied in several different scenarios, yielding not only optimal expected-time leader election in the model without collision detection (Theorem 9), but also to optimal or almost optimal leader election algorithms for multi-hop radio networks with collision detection (Theorem 12) and for directed networks with high probability bound (Theorem 10).

2 Basic primitives

Our algorithms rely on several basic primitives, all being rather standard in the area of distributed computing in radio networks. When called, these subroutines will be implicitly passed an input set S of “sources”. While this set is not common knowledge, and so cannot be an explicit parameter, we will ensure that each node knows whether it is itself a member of S , and so can behave accordingly. We also remark on the synchronization of the algorithm: all calls to these subroutines take some fixed predetermined number of time steps which (assuming knowledge of n and D , or at least common linear upper bounds) is known to all nodes. Therefore, nodes can perform these tasks in a synchronized fashion.

2.1 Model without collision detection

We begin by introducing the primitives we will use in the radio network model without collision detection.

2.1.1 Decay

The Decay protocol, first introduced by Bar-Yehuda et al. [3], is a fundamental primitive employed by many randomized radio network communication algorithms. Its aim is to ensure that, if a node has on or more in-neighbors which wish to transmit a message, it will hear at least one of them.

Algorithm 1 DECA $\text{Y}(S)$ at a node v

```

if  $v \in S$  then
  for  $i = 1$  to  $\log n$ , in time step  $i$  do:  $v$  transmits its ID with probability  $2^{-i}$ 
end if

```

The following lemma (cf. [3]) describes a basic property of DECA Y , as used in our analysis.

Lemma 1 *After four rounds of DECA $\text{Y}(S)$, a node v with at least one in-neighbor in S receives a node ID with probability greater than $\frac{1}{2}$.* \square

While Decay has a very localized effect, to achieve global tasks we will need more complex primitives.

2.1.2 Partial multi-broadcast

We consider a scenario when one wants to broadcast information to the entire network from multiple-sources. We will require a PARTIAL MULTI-BROADCAST(S, f, l) algorithm with the following properties:

- $S \subseteq V$ is a (possibly empty) set of *source nodes*;
- $f : S \rightarrow \{0, 1\}^l$, where $l = O(\log n)$, is a function giving each source a *bit-string to broadcast*. In our applications, this will either be a node ID, or a single bit “1”;
- Each node v interprets some bit-string $m(v)$ upon completion;
 - If $S = \emptyset$, then $m(v) = \epsilon$ (the empty string) for all v ;
 - If $S \neq \emptyset$, then $\forall v \in V \exists s \in S$ with $m(v) = f(s)$, i.e., each node interprets some source’s bit-string.

The broadcasting algorithm of Czumaj and Rytter [8], performed with every node in S operating as a single source and with nodes interpreting the first transmission they receive to be their $m(v)$, yields:

Lemma 2 *There is a Partial Multi-Broadcast algorithm running in $O(D \log \frac{n}{D} + \log^2 n)$ time, which succeeds with high probability.* \square

2.1.3 Selection

We now consider another basic primitive, **SELECTION**, that relies on a combination of **Decay** and **Partial Multi-Broadcast**. The purpose of **SELECTION**, when run with a set of *source nodes* (also called *candidates*) as implicit input, is to determine which of the following three cases hold: if there are 0, 1, or multiple source nodes. In the latter case, it also gives information allowing the removal of the minimum candidate (by ID).

Formally, we require **SELECTION**(S) with the following properties:

- $S \subseteq V$ is a (possibly empty) set of *source nodes*;
- Each node outputs a string $m(v)$ and a bit b :
 - If $S = \emptyset$, then $m(v) = \epsilon$ (the empty string) for all v , and $b = 0$;
 - If $|S| = 1$, then $m(v)$ is equal to the ID of the sole node in S for all v , and $b = 1$;
 - If $|S| > 1$, then $b = 0$ and $m(v)$ is the ID of some node in $S \setminus \{s\}$, where s is the node in S with lowest ID. (Here, we allow that $m(v) \neq m(u)$ for distinct v, u .)

The output bit b is to indicate whether exactly one candidate remains, in which case we would have completed leader election and can terminate. This final condition is to allow the source node with the minimum ID to be removed: if any candidate who receives back a higher ID than his own drops out, then the condition ensures that at least one (min-ID) must drop out, and at least one (max-ID) remains.

We achieve the properties by Algorithm 2. Intuitively, in **SELECTION**(S), candidates broadcast out their IDs, ensuring that every node hears an ID. The executions of **Decay** then ensure that, with high probability, if a node has any in-neighbors who received a different ID, it will hear at least one of these IDs from them. Nodes who did detect such a conflict, called witnesses, then broadcast this information, along with the highest ID they heard, back throughout the network. Since any node ID which is broadcast back in this way must have been higher than a competing ID, the lowest cannot be among them.

Algorithm 2 **SELECTION**(S) at a node v

```

 $m(v) \leftarrow \text{PARTIAL MULTI-BROADCAST}(S, \text{ID}, 16 \log n)$ 
if  $m(v) \neq \epsilon$  then
  for  $i = 1$  to  $16 \log n$  do
    let  $v \in S_i$  if the  $i^{\text{th}}$  bit of  $v$ 's message  $m(v)_i$  is 1
    perform four rounds of DECAY( $S_i$ )
    if  $v$  receives a node ID but  $m(v)_i = 0$  then
       $v$  becomes a member of the set  $W$  of witnesses
       $m(v) \leftarrow \max(m(v), \text{received ID})$ 
    end if
  end for
   $p(v) \leftarrow \text{PARTIAL MULTI-BROADCAST}(W, m, 16 \log n)$ 
  if  $p(v) = \epsilon$  then  $v$  receives output  $(m(v), 1)$ , procedure terminates
  else  $v$  receives output  $(p(v), 0)$ , procedure terminates
else
   $v$  receives output  $(m(v), 0)$ , procedure terminates
end if

```

We first prove a claim needed to detect multiple candidates and then show the correctness of Algorithm 2.

Claim 3 *If Algorithm 2 is run with $|S| > 1$, then the set W of witnesses is non-empty with high probability.*

Proof. For any distinct $u, v \in S$ let $p_{u,v}$ be the probability that there are fewer than $\log n$ bits i such that $m(u)_i = 1$ and $m(v)_i = 0$ or vice versa. Since, on each bit, this occurs independently with probability $\frac{1}{2}$, by a Chernoff bound $p_{u,v} \leq \frac{1}{n^3}$. By the union bound, the probability that any pair has the property is at most $\sum_{u,v \in S} p_{u,v} \leq \frac{|S|^2}{n^3} \leq \frac{1}{n}$. So, with high probability, for every $u, v \in C$ there are at least $\log n$ bits i where $m(u)_i$ and $m(v)_i$ differ.

Assuming that $\text{PARTIAL MULTI-BROADCAST}(C, \text{ID}, 12 \log n)$ is successful (which happens w.h.p.), there will be at least one node in the graph v which has at least one in-neighbor u such that $m(v) \neq m(u)$. Then, w.h.p. there are at least $\log n$ bits i with $m(u)_i = 1$ and $m(v)_i = 0$. If $\text{DECAY}(S_i)$ succeeds on any of these i then v becomes a witness. This happens with probability greater than $1 - (\frac{1}{2})^{\log n} = 1 - \frac{1}{n}$. \square

Lemma 4 *Algorithm 2 satisfies the conditions for SELECTION with high probability, and terminates within $O(D \log \frac{n}{D} + \log^2 n)$ time-steps.*

Proof. If $S = \emptyset$ then clearly $m(v) = \epsilon$ and $b = 1$ for all v . If $|S| = 1$ then w.h.p. all nodes will receive the ID of the sole source, and so none will become witnesses and all will output $b = 1$ as required. If $|S| > 1$ then by the previous claim there will be at least one witness. Witnesses receive at least two IDs (one in the initial $\text{PARTIAL MULTI-BROADCAST}$ and one from the decay phase), and send out the highest of these in the second $\text{PARTIAL MULTI-BROADCAST}$. Assuming these broadcasts are successful, all nodes will then output some non-minimum ID and $b = 0$.

Running time is dominated by that of two rounds of $\text{PARTIAL MULTI-BROADCAST}$, taking $O(D \log \frac{n}{D} + \log^2 n)$ time, and $64 \log n$ rounds of DECAY , taking $O(\log^2 n)$ time. \square

2.1.4 Search

During the course of our high-probability algorithm we will wish to identify an ID or range of IDs which we know to be held by at least one candidate. We can do this by performing a binary search over the range of ID, using $\text{Partial Multi-Broadcast}$ at every step to allow all nodes to agree on each bit of the ID. If IDs are unique and we identify one particular one, then we have performed leader election. However, doing so using this binary search technique would be too slow ($\log n$ times broadcasting time). What we will do, without incurring excessive time-cost, is partially complete the process, and have all nodes agree on the first ℓ bits of an ID, for some parameter ℓ . Since the remaining bits are undecided, this leaves a range of possible IDs.

Accordingly, we define a procedure $\text{SEARCH}(S, \ell)$ with the following properties:

- $S \subseteq V$ is a nonempty set of source nodes;
- Each source node has an ID;
- Each node receives as output the first ℓ bits of the highest ID in use.

These requirements are achieved by Algorithm 3.

Lemma 5 *Algorithm 3 performs Search in $O(\ell(D \log \frac{n}{D} + \log^2 n))$ time, succeeding with high probability.*

Proof. We prove by induction that, after iteration i of the For loop, all nodes know the first i bits of the largest ID in use. For the base case $i = 1$, all nodes with a first bit of 1 will participate in the $\text{PARTIAL MULTI-BROADCAST}$. If there are any such nodes, the whole network will receive a transmission, and know that the first bit is 1; otherwise, all nodes will interpret ϵ and know that it is 0.

Algorithm 3 SEARCH(S, ℓ) at a node v

```
initialise  $m(v)$  as an  $\ell$ -bit string
for  $i = 1$  to  $\ell$  do
  if  $\text{ID}(v)_j = m(v)_j$  for all  $j < i$  and  $\text{ID}(v)_i = 1$  then  $v$  becomes a member of the set  $S$ 
   $p(v) \leftarrow \text{PARTIAL MULTI-BROADCAST}(S, 1, 1)$ 
  if  $p(v) = \epsilon$  then  $m(v)_i = 0$ 
  else  $m(v)_i = 1$ 
end for
 $v$  receives output  $m(v)$ , procedure terminates.
```

The same argument holds for the inductive steps: if the claim is true up to round j , then in round $j + 1$ all nodes whose IDs are in agreement with the highest for the first j bits, and who have a 1 as the $j + 1^{\text{th}}$, participate in the PARTIAL MULTI-BROADCAST. If there are any such nodes, all nodes are aware of this and set the $j + 1^{\text{th}}$ bit to 1, otherwise they set it to 0. Thereby all nodes now agree on the first $j + 1$ bits of the highest ID.

The procedure fails only if one of the $O(\log n)$ calls to PARTIAL MULTI-BROADCAST fails, and so there is high probability of success.

Running time is ℓ times that of PARTIAL MULTI-BROADCAST, i.e. $O(\ell(D \log \frac{n}{D} + \log^2 n))$. \square

2.2 Beep Model

The beep model requires slightly different methods, since messages larger than one bit cannot be transmitted within a single time-step as in radio networks.

2.2.1 Beep Waves

We will need to have an analogue of PARTIAL MULTI-BROADCAST for the beep model. For this purpose we will make use of a standard technique known as *beep-waves*, first mentioned in [10]. Specifically, we require procedure BEEP-WAVE(S, f, ℓ) which satisfies the following:

- $S \subseteq V$ is a (possibly empty) set of source nodes;
- $f : S \rightarrow \{0, 1\}^\ell$, where $\ell = O(\log n)$, is a function giving each source a bit-string to broadcast.
- Each node interprets a string $m(v)$;
 - If $S = \emptyset$, then $m(v) = \epsilon$ (the empty string) for all v ;
 - If $S = \{s\}$, for some $s \in V$, then $\forall v \in V, m(v) = f(s)$;
 - If $|S| > 1$, then for all $v, m(v) \neq \epsilon$. Furthermore, there exists $w \in V$ and two distinct $u, v \in S$ (we allow $w \in \{u, v\}$) such that $m(w) = f(u) \vee f(v) \vee m$, for some bit-string m .

The last condition may seem convoluted; the reason for it is that, while we cannot guarantee messages are correctly received as in the single source case, we will at least need some means of telling that there were indeed multiple sources. This will be detailed later, but for now we require that, as well as all nodes receiving some non-empty message, at least one receives the logical **OR** of two source messages, possibly with some extra 1s. This node will act in a similar way to the *witnesses* in the SELECTION algorithm.

We achieve these conditions with Algorithm 4. The intuition behind the algorithm is that a source uses beeps and silence to transmit the 1s and 0s of its message respectively (prefixed by a 1 so that it is obvious

when transmission starts), and other nodes forward this pattern, one adjacency layer per time-step, by simply relaying a beep when they hear one. This process takes $O(D + \ell)$ time, where ℓ , as introduced above, is the maximum length, in bits, of message that is to be broadcast. Since we are here making the restriction $\ell = O(\log n)$, our running time is $O(D + \log n)$.

Algorithm 4 BEEP-WAVE(S, f, ℓ) at a node v

```

initialize  $m(v)_i = 0 \forall i \in [\ell]$ 
if  $v \in S$  then
     $v$  beeps in time-step 0
    for  $i = 1$  to  $\ell$  do
        if the  $i^{\text{th}}$  bit of  $f(v)$  is 1 then  $v$  beeps in time time-step  $3i$  and we set  $m(v)_i = 1$ 
        else  $v$  receives a beep in time time-step  $3i$  and we set  $m(v)_i = 1$ 
    end for
else
    Let  $j$  be the time-step in which  $v$  receives its first beep
     $v$  beeps in time-step  $j + 1$ 
    for  $i = 1$  to  $\ell$  do
        if  $v$  receives a beep in time-step  $j + 3i$  then  $v$  beeps in time-step  $j + 3i + 1$  and we set  $m(v)_i = 1$ 
        else set  $m(v)_i = 1$ 
    end for
end if

```

We prove that the algorithm has the desired behavior:

Lemma 6 *If BEEP-WAVE(S, f, ℓ) is run with $S = \{s\}$, then $\forall v \in V, m(v) = f(s)$*

Proof. Partition all nodes into layers depending on their distance from the source s , i.e. layer $L_i = \{v \in V : \text{dist}(v, s) = i\}$. We first note that a node in layer i beeps for the first time in time-step i , since this first beep will propagate through the network one layer per time-step. The algorithm then ensures that such a node will beep only in time-steps equivalent to $i \bmod 3$, and only if a beep was heard in the previous step. Since all neighbors of the node must be in layers $i - 1, i$, and $i + 1$, only messages from neighbors in layer $i - 1$ can be relayed (as these are the only neighbors whose beeps are in time-steps equivalent to $i - 1 \bmod 3$). It is then easy to see that layers act in unison, and beep if and only if the previous layer beeped in the previous time-step. \square

Lemma 7 *If BEEP-WAVE(S, f, ℓ) is run with $|S| > 1$ then there exists $w \in V$ and two distinct $u, v \in S$ (we allow $w \in \{u, v\}$) such that $m(w) = f(u) \vee f(v) \vee m$, for some bit-string m .*

Proof. Let u, v be the closest pair of sources in the graph. Let w be the midpoint on the shortest $u \rightarrow v$ path (if the path is of odd length, pick either midpoint arbitrarily).

If w is a source, then we can assume, without loss of generality, that $w = u$ and v is an adjacent source. Then if $f(w)_i = 1$ or $f(v)_i = 1$, w receives or transmits a beep in time-step $3i$ and sets $m(w)_i = 1$, so we are done.

Otherwise, assume without loss of generality that $\text{dist}(w, u) \leq \text{dist}(w, v) \leq \text{dist}(w, u) + 1$, and denote $j := \text{dist}(w, u) - 1$. w receives its first beep in time-step j . Then, since u is the closest source to every node along the shortest $u \rightarrow w$ path and v is the closest source to every node along the shortest $v \rightarrow w$ path, beeps from u and v will always be relayed along these paths. So, if $f(u)_i = 1$, w receives a beep in time-step $j + 3i$, and if $f(v)_i = 1$, w receives a beep in time-step $\text{dist}(v, w) - 1 + 3i = j + 3i$ or $j + 3i + 1$

(unless w beeps itself in time-step $j + 3i + 1$, i.e. it received a beep in time-step $j + 3i$). In either case, $m(w)_i$ is set to 1. \square

Lemma 8 *Algorithm 4 correctly achieves the Beep-Wave conditions in $O(D + \ell)$ time-steps.*

Proof. Correctness for the cases $|S| = 1$ and $|S| > 1$ follow from Lemmas 6–7, and the case $S = \emptyset$ follows since no node ever beeps. To analyze the running time: clearly all sources will have ceased transmission after $O(\ell)$ time, and since beeps are propagated through the network one layer per time-step, it may be a further D time-steps before a source’s last beep is heard by the whole network, yielding $O(D + \ell)$ time. \square

3 Leader election algorithms

Having defined all of the primitives needed, we can now present our leader election algorithms.

3.1 Optimal expected-time leader election in radio networks

The approach in our optimal expected-time leader election, Algorithm 5, follows the ideas developed by Bar-Yehuda et al. [2], which in our framework can be described as follows: We repeatedly attempt to randomly select a single candidate, and then run a process to check whether we have indeed done so. If we have, we terminate the algorithm, and if not, we continue. Since we can achieve a constant probability of selecting one candidate, the expected number of iterations required to do so is also constant.

Algorithm 5 Leader Election (n) at a node v , expected time

```

loop
   $v$  chooses to be in the set  $C$  of candidates with probability  $\frac{1}{n}$ 
  if  $v \in C$  then  $v$  chooses a  $16 \log n$ -bit ID independently and uniformly at random
    perform  $(m(v), b) \leftarrow \text{SELECTION}(C)$ 
    if  $b = 1$  then  $v$  outputs  $m(v)$ , algorithm terminates
end loop

```

Theorem 9 *Algorithm 5 terminates in expected $O(D \log \frac{n}{D} + \log^2 n)$ time and correctly performs leader election with high probability.*

Proof. We first prove the correctness of Algorithm 5 and then analyze its running time.

Correctness: If, in any iteration, $|C| = 1$, then w.h.p., $\text{SELECTION}(C)$ will be successful (i.e. every node will receive the ID of the one candidate and $b = 1$), and the algorithm will terminate.

If $|C| = 0$ or $|C| > 1$, then w.h.p. $\text{SELECTION}(C)$ will correctly cause every node to set $b = 0$ and therefore again the iteration will end and the algorithm will continue.

Running time: In any particular iteration, the probability that $|C| = 1$ is $n \cdot \frac{1}{n} \cdot (\frac{n-1}{n})^{n-1} \geq \frac{1}{e}$. Therefore the expected number of iterations required until $|C| = 1$ is fewer than 3.

The running time of each iteration is dominated by that of SELECTION , which takes $O(D \log \frac{n}{D} + \log^2 n)$ time. Hence, since an expected constant number of iterations is required, the expected asymptotic running time of the whole algorithm is $O(D \log \frac{n}{D} + \log^2 n)$. \square

Observation on simulation of single-hop networks We note that our approach in Algorithm 5 is similar to that used by Bar-Yehuda et al. [2] to simulate a single time-step of a single-hop network with collision detection using $O(T_{BC})$ time-steps in a multi-hop network without collision detection (where T_{BC} is the number of time-steps required to broadcast a single message throughout the latter network). They cite the leader election algorithm of Willard [18], for single-hop networks, as a prime application of their simulation method. This algorithm takes $O(\log \log n)$ time-steps and is optimal under its assumptions about the model. However, the standard model of multi-hop networks assumes that nodes have knowledge of n , or at least a common linear upper bound; this assumption is made, for example, in the work of Ghaffari and Haeupler [10], who present leader election algorithms with the fastest-known high-probability running times. If this knowledge is available, a simpler algorithm in single-hop networks is possible (Algorithm 6), which achieves successful transmission, and hence election of a leader, in expected constant time.

From the perspective of this work, Algorithm 5 can be understood as being equivalent to the simulation of Algorithm 6, using the optimal broadcasting algorithm of Czumaj and Rytter [8].

Algorithm 6 Leader Election in single-hop networks with collision detection

```

loop
  each node independently chooses to be a candidate with probability  $\frac{1}{n}$ 
  each candidate randomly chooses a  $\log n$ -bit ID
  each candidate transmits its ID
  if a candidate succeeded in transmitting then that candidate is designated leader; algorithm terminates
end loop

```

While this approach yields an optimal expected-time algorithm, if we wished to achieve high probability the $\Omega(\log n)$ lower bound for single-hop networks would seem to limit the effectiveness a similar approach. However, due to the fact that SELECTION can not only inform the network that there are multiple candidates but also provide enough information for the lowest to drop out, we can employ our techniques in a way that is not equivalent to simulating single-hop networks. As a result, we can get a high-probability leader election algorithm which beats the $O((D \log \frac{n}{D} + \log^2 n) \cdot \log n)$ simulation lower bound.

3.2 Radio networks, high-probability running time

We present our second leader election algorithm for radio networks, which terminates in $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ time and succeeds with high probability (Algorithm 7). This is the fastest algorithm for high probability of success on directed networks, though for undirected networks, it is slower than that of [10].

The idea is to trade-off between the two methods of cutting down the field of candidates: we first perform a partial binary search to find a range of IDs in which we know that, with high probability, there are between 1 and $3\sqrt{\log n}$ candidates. We then remove candidates one at a time using our SELECTION procedure.

Theorem 10 *Algorithm 7 completes leader election in time $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ with high probability.*

Proof. By Chernoff bound $\log n \leq |C| \leq 16 \log n$ w.h.p. Conditioning on this, we prove that with high probability no $3\sqrt{\log n}$ candidates from C share the same first $\sqrt{\log n}$ bits of their ID. Let \mathcal{E} be the event

Algorithm 7 Leader Election (n) at a node v , high probability

v chooses to be in the set C of **candidates** with probability $\frac{4 \log n}{n}$
if $v \in C$ **then** v chooses a $\log n$ -bit ID independently and uniformly at random
 $m(v) \leftarrow \text{SEARCH}(C, \sqrt{\log n})$
if $m(v) \neq$ the first $\sqrt{\log n}$ bits of $\text{ID}(v)$ **then** v removes itself from C
loop $3\sqrt{\log n}$ times
 perform $(p(v), b) \leftarrow \text{SELECTION}(C)$
 if $b = 1$ **then** v outputs $p(v)$, algorithm terminates
 else if $p(v) > \text{ID}(v)$ **then** v removes itself from C
end loop

that this does not occur, i.e., that there are $3\sqrt{\log n}$ candidate such candidates. Then, by a union bound,

$$\begin{aligned} \Pr[\mathcal{E}] &\leq \sum_{C' \subseteq C \text{ such that } |C'| = 3\sqrt{\log n}} \Pr[\text{candidates in } C' \text{ share the first } \sqrt{\log n} \text{ bits of their ID}] \\ &\leq \binom{16 \log n}{3\sqrt{\log n}} \cdot 2^{-(\sqrt{\log n})(3\sqrt{\log n}-1)} \leq (6e\sqrt{\log n})^{3\sqrt{\log n}} \cdot 2^{-(\sqrt{\log n})(3\sqrt{\log n}-1)} \leq n^{-2}. \end{aligned}$$

Therefore, after execution of **SEARCH**, with high probability no more than $2\sqrt{\log n}$ (and no fewer than 1) candidates remain. While there are still multiple candidates, each iteration of **SELECTION** removes at least one (the one with lowest ID), and so after $2\sqrt{\log n}$ iterations, only one remains. Since for any constant c we can bound the probability of failure of **SELECTION** above by n^{-c} at only a constant factor slowdown, we can easily ensure that all $2\sqrt{\log n}$ succeed w.h.p.

The running time of the algorithm is dominated by the $O((D \log \frac{n}{D} + \log^2 n)\sqrt{\log n})$ time of **SEARCH**, and the $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ total time of $2\sqrt{\log n}$ iterations of **SELECTION**. This gives a total running time of $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$. \square

3.3 Beep Model

Communication in the beep model relies on some different techniques from those used in radio networks, since messages cannot be transmitted in a single time-step and must instead be carried by procedures such as **BEEP-WAVE**. Nonetheless, our algorithm for leader election in the beep model (Algorithm 8) bears some similarities to Algorithm 5. Again, we repeatedly attempt to randomly select a single candidate, and then run a procedure to check whether we have successfully done so. This is complicated by the fact that our method of broadcast experiences interference when attempted from multiple sources.

Claim 11 *In any iteration of the loop of Algorithm 8, w.h.p., every candidate chooses a different ID.*

Proof. First, we note that $|C| \leq \log n$ w.h.p. Next, conditioned on $|C| \leq \log n$, by the union bound, the probability that any pair of candidates a, b chooses the same ID is at most $\sum_{a,b \in C} \frac{1}{\binom{4 \log n}{\log n}} \leq \log^2 n \cdot \frac{1}{4^{\log n}} = \frac{\log^2 n}{n^2}$. This implies that every candidate chooses a different ID w.h.p. \square

Theorem 12 *Algorithm 8 terminates in expected $O(D + \log n)$ time and correctly performs leader election with high probability.*

Algorithm 8 Leader Election in beep model (n) at a node v

```
loop
   $v$  chooses to be in the set  $C$  of candidates with probability  $\frac{1}{n}$ 
  if  $v \in C$  then  $v$  chooses a  $4 \log n$ -bit ID with exactly  $\log n$  1s uniformly at random
  perform  $m(v) \leftarrow \text{BEEP-WAVE}(C, \text{ID}, 4 \log n)$ 
  if  $m(v) \neq \epsilon$  then
    if  $m(v)$  contains more than  $\log n$  1s then
       $v$  becomes a member of the set  $W$  of witnesses
      perform  $p(v) \leftarrow \text{BEEP-WAVE}(W, 1, 1)$ 
      if  $p(v) = \epsilon$  then  $v$  outputs  $m(v)$ , algorithm terminates
    end if
  end if
end loop
```

Proof. We first prove the correctness of Algorithm 8 and then analyze its running time.

Correctness: In each iteration of the loop within Algorithm 8, we consider three possible cases for the size of the set of candidates C :

If $|C| = 1$, then $\text{BEEP-WAVE}(C, \text{ID}, 4 \log n)$ will ensure that every node receives the ID of the single candidate. Since this will have exactly $\log n$ **1s**, The set W of witnesses will be empty, so the algorithm will successfully terminate.

If $|C| = 0$, then all nodes will receive the empty string ϵ during $\text{BEEP-WAVE}(C, \text{ID}, 4 \log n)$, and therefore the iteration will end and the algorithm will continue.

If $|C| > 1$, then by Lemma 8 there exists $w \in V$ and distinct $u, v \in S$ such that $m(w) = f(u) \vee f(v) \vee m$, for some bit-string m . Since w.h.p. $f(u) \neq f(v)$, and both contain $\log n$ **1s**, $m(w)$ contains strictly more than $\log n$ **1s**, and so w becomes a witness. Then each node receives a transmission during $\text{BEEP-WAVE}(W, 1, 1)$, so again the iteration will end and the algorithm will continue.

Running time: In any particular iteration, the probability that $|C| = 1$ is $n \cdot \frac{1}{n} \cdot (\frac{n-1}{n})^{n-1} \geq \frac{1}{e}$. Therefore the expected number of iterations required until $|C| = 1$ is fewer than 3.

The running time of each iteration is dominated by that of $\text{BEEP-WAVE}(C, \text{ID}, 4 \log n)$ taking $O(D + \log n)$ time. Hence, the expected asymptotic running time of the whole algorithm is $O(D + \log n)$. \square

4 Conclusion

In this paper we present the first asymptotically optimal expected time leader election algorithms: one for the setting without collision detection, and another for undirected graphs with collision detection. Our leader election algorithm in networks without collision detection follows the simulation approach of Bar-Yehuda et al. [2] and achieves an optimal $O(D \log \frac{n}{D} + \log^2 n)$ expected running time for directed (and also undirected) networks, when the size of the network, or a common upper bound, is known. We then extend the ideas behind this algorithm to take advantage of the presence of collision detection (even in the simplest beep model), and present an algorithm for the corresponding model which terminates in optimal $O(D + \log n)$ expected time, this time though only for undirected networks. We also give an algorithm which for radio networks which terminates in $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ time and succeeds with high probability.

There are several important open question in this area. Firstly, can anything be done in the case of directed graphs with collision detection? Here, the method of beep-waves used as the backbone of Algorithm 8 does not provide the same guarantees, as a wave can revisit nodes it already passed through and be

interpreted as a new transmission. The best known leader election algorithm in this setting is Algorithm 5 with an expected running time of $O(D \log \frac{n}{D} + \log^2 n)$, but the only known lower bound is the same as that of undirected graphs, $\Omega(D + \log n)$, and we would wish to close this gap.

Secondly, what is the complexity of the leader election problem if we require high probability bounds for the running time? The best results in this direction, for undirected graphs, are the algorithms of Ghaffari and Haeupler [10], with running times of $O(D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log \log n, \log \frac{n}{D}\}$ without collision detection, and $O(D + \log n \log \log n) \cdot \min\{\log \log n, \log \frac{n}{D}\}$ with collision detection, but these running times are off from the respective lower bounds by both an additive term and multiplicative factor. For the directed case our $O((D \log \frac{n}{D} + \log^2 n) \cdot \sqrt{\log n})$ -time algorithm is the best known, but is again slower than the $O(D \log \frac{n}{D} + \log^2 n)$ lower bound by a multiplicative factor.

References

- [1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2): 290–298, October 1991.
- [2] R. Bar-Yehuda, O. Goldreich, and A. Itai. Efficient emulation of single-hop radio network with collision on multi-hop radio network with no collision detection. *Distributed Computing*, 5: 67–71, 1991.
- [3] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1): 104–126, August 1992.
- [4] B. S. Chlebus, D. R. Kowalski, and A. Pelc. Electing a leader in multi-hop radio networks. In *Proceedings of the 16th International Conference on Principles of Distributed Systems (OPODIS)*, pages 106–120, 2012.
- [5] M. Chrobak, L. Gąsieniec, and W. Rytter. A randomized algorithm for gossiping in radio networks. In *Proceedings of the 7th Annual International Computing Combinatorics Conference (COCOON)*, pages 483–492, 2001.
- [6] A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcasting in radio networks of unknown topology. *Theoretical Computer Science*, 302(1–3): 337–364, June 2003.
- [7] A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. *Proceedings of the 24th International Symposium on Distributed Computing (DISC)*, pages 148–262, 2010.
- [8] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 492–501, 2003.
- [9] K.-T. Förster, J. Seidel, and R. Wattenhofer. Deterministic leader election in multi-hop beeping networks. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 212–226, 2014.
- [10] M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–766, 2013. Also in arXiv 1210.8439, version 2, April 2014.

- [11] M. Ghaffari, N. Lynch, and S. Sastry. Leader election using loneliness detection. *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 268–282, 2011.
- [12] A. G. 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, July 1985.
- [13] T. Jurdziński and G. Stachowiak. Probabilistic algorithms for the wake-up problem in single-hop radio networks. *Theory of Computing Systems*, 38(3): 347–367, May 2005.
- [14] D. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. In *Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 73–82, 2003.
- [15] D. R. Kowalski and A. Pelc. Leader election in ad hoc radio networks: A keen ear helps. *Journal of Computer and System Sciences* 79(7): 1164–1180, November 2013.
- [16] E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3): 702–712, June 1998.
- [17] K. Nakano and S. Olariu. Uniform leader election protocols for radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(5): 516–526, May 2002.
- [18] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal on Computing*, 15(2): 468–477, May 1986.