Leader Election Using Loneliness Detection[⋆]

Mohsen Ghaffari, Nancy Lynch, and Srikanth Sastry

CSAIL, MIT
Cambridge, MA 02139, USA

Abstract. We consider the problem of leader election (LE) in single-hop radio networks with synchronized time slots for transmitting and receiving messages. We assume that the actual number n of processes is unknown, while the size u of the ID space is known, but possibly much larger. We consider two types of collision detection: *strong (SCD)*, whereby all processes detect collisions, and *weak (WCD)*, whereby only non-transmitting processes detect collisions.
We introduce *loneliness detection (LD)* as a key subproblem for solving LE in WCD systems. LD informs all processes whether the system contains exactly one process or more than one. We show that LD captures the difference in power between SCD and WCD, by providing an implementation of SCD over WCD and LD. We present two algorithms that solve deterministic and probabilistic LD in WCD systems with time costs of $\mathcal{O}(\log \frac{u}{n})$ and $\mathcal{O}(\min(\log \frac{u}{n}, \frac{\log(1/\epsilon)}{n}))$, respectively, where ϵ is the error probability. We also provide matching lower bounds.
We present two algorithms that solve deterministic and probabilistic LE in SCD systems with time costs of $\mathcal{O}(\log u)$ and $\mathcal{O}(\min(\log u, \log\log n + \log(\frac{1}{\epsilon})))$, respectively, where ϵ is the error probability. We provide matching lower bounds.

1 Introduction

We study the leader election problem in single-hop radio networks with synchronized time slots for transmitting messages, but where messages are subject to collisions. We focus on the time cost of electing a leader and the dependence of this cost on the number n of actual processes in the network as well as on a known, finite ID space I of the processes. We assume that while n may be unknown, each process knows its own ID and the ID space I. We only restrict the size of the ID space, $u = |I|$, to be at least n; the number of processes in the system may be much smaller than the size of the ID space.

The time cost of leader election depends significantly on the ability of the processes to detect message collisions. The problem has been well studied in single-hop systems which have no collision detection (*e.g.*, [3, 6, 10, 1, 8]) and in systems with *strong collision detection* (SCD) in which all processes can detect message collisions (*e.g.*, [11, 2, 5, 3, 7, 9]). However, the cost of leader election is

⋆ This work partially supported by NSF under award numbers CCF-0937274, CCF-0939370, and CCF-0726514, and by AFOSR under award number FA9550-08-1-0159.

under-explored in systems with *weak collision detection* (WCD) wherein only the listening processes detect message collisions. We focus on the time costs of solving leader election, both deterministic and randomized, in WCD systems and compare them to the time costs for leader election in SCD systems.

The primary challenge to solving leader election in WCD systems is to distinguish the following two cases: (1) $n > 1$ and all the processes transmit simultaneously resulting in a collision, which remains undetected because no process is listening, (2) $n = 1$ and any message transmitted by the process does not collide, but the successful transmission is undetected because no process is listening. In both cases the transmitting processes receive the same feedback from the WCD system despite different outcomes. Note that these two cases are distinguishable in SCD systems. Hence, *loneliness detection* — determining whether or not there is exactly one process in the system — is a key subproblem of leader election.

Summary of Results. We define the Loneliness Detection (LD) problem in Sect. 4 and determine the time complexity of solving LD in single-hop wireless networks in Sect. 5. We show that LD can be solved deterministically in WCD systems in $\mathcal{O}(\log \frac{u}{n})$ time slots for $n > 1$ and in $\mathcal{O}(\log u)$ time slots for $n = 1$. Interestingly, in the probabilistic case, if $n > 1$, LD can be solved in WCD systems in a constant number of rounds with high probability; however, if $n = 1$, then our algorithm takes $\mathcal{O}(\log u)$ time slots. We demonstrate that these time bounds are tight by presenting matching lower bounds.

In Sect. 4, we implement an SCD system on top of a WCD system augmented with a solution to LD. This allows us to deploy SCD-based protocols on WCD systems. We explore such SCD-based protocols for LE in Sect. 6 where we present upper and lower bounds for both deterministic and randomized LE in SCD systems. First, we present a deterministic LE protocol that terminates in at most $\mathcal{O}(\log u)$ time slots and show a matching lower bound. For probabilistic LE, we interleave Nakano and Olariu's algorithm from [9] with our deterministic algorithm to solve the problem in $\mathcal{O}(\min(\log u, \log\log n + \log(\frac{1}{\epsilon}))$ rounds with termination probability at least $1-\epsilon$ (where $1 < \epsilon < 0$). We present a lower bound of $\Omega(\min(\log(\frac{u}{n}), \log(\frac{1}{\epsilon})))$ for probabilistic LE on SCD systems with termination probability at least $1 - \epsilon$. Note that the lower and upper bounds match when $\epsilon = \mathcal{O}(\frac{1}{n})$. Subsequently, in Sect. 6, we demonstrate that the same upper and lower bounds hold for LE in WCD systems as well.

The full proofs omitted here due to space constraints are available in [4].

2 System Models, Definitions, and Notations

Our model considers a finite set of n processes with unique IDs from I, a finite ID space of size u. The set $J \subseteq I$ denotes the set of IDs of the n processes.

Processes. Processes communicate by broadcasting messages from a fixed alphabet $\mathcal{M}$ on the shared channel. We assume that $\mathcal{M}$ does not contain the special placeholder elements $\perp$ and $\top$, which denote silence and collision, respectively. We assume that time is divided into *rounds*, and processes have synchronized clocks and can detect the start and end of each round. Processes transmit only at round boundaries, and each transmission is contained within a single round. We assume that all processes wake up at time 0, which is the start of round 1.

A process i transmits a message m in round r through the action $\mathit{send}(m, r)_i$ and receives a message m' in round r through the action $\mathit{receive}(m', r)_i$. If a process i does not send a message in round r, then, we say that process i executes action $\mathit{send}(\perp, r)_i$. If a process does not send a message in round r, then it is assumed to be listening in round r. If a process i does not receive a message in round r, then the process either receives silence through action $\mathit{receive}(\perp, r)_i$ or receives a collision notification through action $\mathit{receive}(\top, r)_i$. We assume that in every execution, for each process i and each round r, exactly one event of the form $\mathit{send}(*, r)_i$ and exactly one event of the form $\mathit{receive}(*, r)_i$ occurs.

Wireless Channels. A wireless channel is a broadcast medium in which at most one process can successfully send a message in any round. We assume that a known, nondecreasing *time-bound function* $b : \mathbb{N}^+ \rightarrow \mathbb{R}^{\geq 0}$, which maps each round to an upper bound on the real time at which the round ends. Any channel that satisfies a time-bound function b is said to be a *b-time-bounded channel*.

The behavior of a channel in a round r is determined by the set T of transmitting processes in round r. If no process transmits a message in round r, *i.e.*, $|T| = 0$, then for each process i in the system, the event $\mathit{receive}(\perp, r)_i$ occurs; that is, all processes receive silence in round r. If exactly one process j transmits a message (say) m in round r, *i.e.*, $|T| = 1$, then for each process i in the system, the event $\mathit{receive}(m, r)_i$ occurs. If two or more processes send messages in a given round, *i.e.*, $|T| > 1$, then we say that the round experiences a message collision. The responses given by a channel in the event of a message collision are determined by their collision detection ability. We consider two types of channels.

Weak Collision Detection (WCD) Channels. In WCD channels, in the case of a collision, every transmitting process receives its own message, and every process that is listening receives $\top$. That is, if $|T| > 1$, then for each process i in T, where event $\mathit{send}(m_i, r)_i$ occurs, the event $\mathit{receive}(m_i, r)_i$ occurs, and for each process i not in T, the event $\mathit{receive}(\top, r)_i$ occurs. We denote the time-bound function of a WCD channel by b_{WCD}.

Strong Collision Detection (SCD) Channels. In SCD channels, if a message collision occurs, then all processes receive $\top$ in that round. That is, if $|T| > 1$, then for each process i, the event $\mathit{receive}(\top, r)_i$ occurs.

We also consider *probabilistic SCD channels* in which the real-time duration of a round is specified by a function $\rho : \mathbb{N}^+ \times [0, 1] \rightarrow \mathbb{R}^{\geq 0}$, where $\rho(r, \epsilon)$ is an upper bound on the real time by which round r terminates with probability at least $1-\epsilon$. We assume that ρ is nondecreasing with respect to r and nonincreasing with respect to ϵ. Additionally, we assume that every round terminates in finite time with probability 1; that is, for each round r, $\rho(r, 0)$ is finite. We say that a probabilistic SCD channel whose round duration is upper bounded by a function ρ is *ρ-time-bounded*.

We assert that systems with SCD are at least as powerful as systems with WCD because SCD provides more information than WCD to the transmitting processes in a given round. A proof appears in [4].

3 The Leader Election Problem

Leader election (LE) is a problem in which each process i eventually outputs $leader(l)_i$ where l is the ID of some process in the system, and the process l is the *leader*. The safety properties of LE state that every process i performs at most one $leader_i$ event, and no two processes output different leaders.

We consider two variants of the LE problem, deterministic and probabilistic, which differ only in their liveness properties. The *deterministic liveness* property states that in any execution, for every process $i \in J$, some event of the form $leader(*)_i$ occurs. The *probabilistic liveness* property states that in the space defined by all infinite executions, for every process i some event of the form $leader(*)_i$ occurs with probability 1. For each process i, an upper bound on the number of rounds within which a $leader_i$ event occurs with probability at least $1-\epsilon$ is given by $\rho_{LE}(\epsilon)$ where $\rho_{LE} : [0,1] \rightarrow \mathbb{N}^+$ is a nonincreasing function. *Deterministic leader election* satisfies the safety properties and the deterministic liveness property, whereas *probabilistic leader election* satisfies the safety properties and the probabilistic liveness property.

For the purposes of demonstrating lower bounds, we also consider the variant η-LE of leader election where $\eta \in \mathbb{N}^+$. Specifically, *deterministic η-LE* denotes the variant of deterministic leader election in which the system consists of η processes and η is known to the processes; similarly, *probabilistic η-LE* denotes the variant of probabilistic leader election in which the system consists of η processes and η is known to the processes.

Prior work. There is a significant body of work exploring LE in wireless systems with collisions. Most of the results focus on LE in SCD systems. For deterministic LE in single-hop SCD systems, there exist matching time bounds $\mathcal{O}(\log n)$ from [2, 5] and $\Omega(\log n)$ from [3] where n, the number of processes in the system, is known. When n is unknown, the best known deterministic upper bound on the time complexity of LE in SCD systems is $\mathcal{O}(n)$ in [7] for arbitrary multi-hop networks of which single-hop is a special case; however, the result in [7] assumes that an upper bound u of n is known and is $\mathcal{O}(n)$. To our knowledge, better upper and lower bounds are not known.

For probabilistic LE in single-hop SCD systems, Willard presents an algorithm in [11] that solves LE on SCD systems in expected time $\mathcal{O}(1)$, $\mathcal{O}(\log\log u)$, and $\log\log n + o(\log\log n)$ in the cases where n is known, where n is unknown, but u is known, and where both n and u are unknown, respectively. For the case where n and u are unknown, the results in [9] provided an improved algorithm with running time $\log\log n + o(\log\log n) + \mathcal{O}(\log(\frac{1}{\epsilon}))$ with probability of termination at least $1-\epsilon$. A lower bound of $\Omega(\log\frac{1}{\epsilon})$ for this problem has been presented in [9] only for "uniform algorithms" in which all the processes transmit with the same probability in each round (although the probability can vary from one round to another).

To our knowledge, the problem of LE seems to be relatively under-explored in the context of WCD systems. The best known time bounds for deterministic LE in single-hop WCD systems, based on the results for broadcast in multi-hop wireless networks in [10], is $\Theta(\log n)$ where n is known.

4 The Loneliness Detection Problem

Loneliness detection (LD) is a service that interacts with processes through output *alone*. In some round r, LD outputs $alone(a, r)_i$ for all processes i where a is Boolean. The safety properties state that if a is *true* there is exactly one process in the system, and if a is false, then there is more than one process. Note that LD outputs its *alone* event at all processes in the same round.

We consider two variants of LD, deterministic and probabilistic, which differ only in their liveness properties. The *deterministic liveness* property states that in any execution, for each process i, some $alone_i$ event occurs. The *probabilistic liveness* property states that in the probability space defined by all infinite executions, for each process i, some $alone_i$ event occurs with probability 1. The upper bound on the number of rounds within which some *alone* event occurs with probability at least $1 - \epsilon$ is given by $\rho_{LD}(\epsilon)$ where $\rho_{LD} : [0, 1] \rightarrow \mathbb{N}^+$ is non-increasing, and $\rho_{LD}(0)$ is finite. A *deterministic loneliness detector* satisfies the safety and the deterministic liveness properties, whereas a *probabilistic loneliness detector* satisfies the safety and the probabilistic liveness properties.

A solution to deterministic LD in which an *alone* event occurs within r_{LD} rounds is said to be a r_{LD}*-round-bounded*, and a solution to randomized LD in which some *alone* event occurs with probability at least $1 - \epsilon$ within $\rho_{LD}(\epsilon)$ rounds is said to be $\rho_{LD}(\epsilon)$*-round-bounded*.

We show that Loneliness Detection (LD) is, in a sense, exactly the difference between SCD and WCD. Note that LD is solved on SCD systems using the following trivial algorithm. Each process i in the system sends a message m at the beginning of round 1 and waits until the end of round 1. If a $\top$ is received at the end of round 1, then the algorithm returns $alone(false, 1)_i$, otherwise it returns $alone(true, 1)_i$.

We now present an algorithm that implements an SCD channel over a WCD system augmented with an LD service. In order to distinguish the actions of the two channels, we rename the *send* and *receive* actions associated with the WCD channel as $sendWCD$ and $receiveWCD$ actions, respectively.

Pseudocode Notation. When an action at a process is triggered by an event e, we denote the trigger with "**upon** e" in the pseudocode. Similarly, when the automaton is waiting for the occurrence of an event e to proceed, we denote it with "**wait until** e". Instances in which an algorithm performs an action a are denoted "**perform** a".

We also use the following notation to bind values to certain variables. Consider the statements "upon $e(x, y)$" and "wait until $e(x, y)$". In both cases, if (say) x is undefined and y is defined at the point in the code where the statements occur, then the semantics of the code is to wait for any event of the form $e(*, y)$, and when an event (say) $e(x', y)$ occurs, bind the value of x' to x.

Algorithm Description. The algorithm consists of two concurrent tasks: Init and Communicate. In the Init task each process i waits for the $alone(a_{LD}, r_{LD})_i$ event from the LD service. The Communicate task consists of two WCD rounds, called Transmit and Ack, which are executed for every round r_s of the SCD channel. Let T denote the set of processes that transmit some message m_i in

Algorithm 1 Implementing SCD on a WCD system using an LD service.

```
Each process $i$ executes two concurrent tasks: Init and Communicate.
Variables:
    $m \in \mathcal{M} \cup \{\perp\}$
    $m', p2msg \in \mathcal{M} \cup \{\top, \perp\}$
    $r_{LD}, r_s \in \mathbb{N}^+$
Task Init:
    Wait until event $alone(a_{LD}, r_{LD})_i$ from the LD service; halt
Task Communicate:
    loop forever
        /* Round $r_s$ for the SCD channel starts here */
        upon $send(m, r_s)_i$ wait for Task Init to terminate
        Transmit Round:
            perform $sendWCD(m, r_{LD} + 2r_s - 1)_i$
            wait until $receiveWCD(m', r_{LD} + 2r_s - 1)_i$
        Ack Round:
            if ($m = \perp$ and $m' \in \mathcal{M}$) then perform $sendWCD(\text{“}ack\text{”}, r_{LD} + 2r_s)_i$
            else perform $sendWCD(\perp, r_{LD} + 2r_s)_i$
            wait until $receiveWCD(p2msg, r_{LD} + 2r_s)_i$
        if ($m' = \perp$) then perform $receive(\perp, r_s)_i$
        else if ($a_{LD} = true$) then perform $receive(m, r_s)_i$
        else if ($p2msg \neq \perp$) then perform $receive(m', r_s)_i$
        else perform $receive(\top, r_s)_i$
        /* Round $r_s$ ends here */
    end loop
```

round r_s of the SCD channel via $send(m_i, r_s)_i$ for each process $i \in T$. In the Transmit round, each process $i \in T$ executes $sendWCD(m_i, r_{LD} + 2r_s - 1)_i$. All other processes listen to the channel via $sendWCD(\perp, r_{LD} + 2r_s - 1)_i$. At the end of the Transmit round, each process $i \in T$ receives its own message via the event $receiveWCD(m', r_{LD} + 2r_s - 1)_i$ where $m' = m_i$. Each listening process j receives either some message, $\perp$, or $\top$ via $receiveWCD(m', r_{LD} + 2r_s - 1)_j$. In the Ack round, each process $i \in T$ listens to the channel via the event $sendWCD(\perp, r_{LD} + 2r_s)_i$. Each process $j \in J \setminus T$ sends an “*ack*” via the event $sendWCD(\text{“}ack\text{”}, r_{LD} + 2r_s)_j$ iff j received a message in the Transmit round; otherwise j listens to the channel via the event $sendWCD(\perp, r_{LD} + 2r_s)_j$. At the end of the Ack round, each process receives either “*ack*”, $\perp$, or $\top$ via $receiveWCD(p2msg, r_{LD} + 2r_s)_*$. If $p2msg$ is “*ack*” or $\top$, then the transmission in the Transmit round was successful, and m' is that transmitted message; so the algorithm outputs $receive(m', r_s)_i$ at each process i. If a_{LD} is $true$, then there is only one process in the system, and so the transmission in the Transmit round was successful, and the algorithm outputs $receive(m, r_s)_i$ at the lone process i. However, if a_{LD} is $false$ and $p2msg$ is $\perp$, then there was a collision, and the algorithm outputs $receive(\top, r_s)_i$ at each process.

Note that although the Init and Communicate tasks are executed concurrently, the Communicate task waits for the Init task to terminate before proceeding to sending and receiving messages on the WCD channel. The pseudocode is shown in Algorithm 1.

Theorem 1. *Algorithm 1 implements a deterministic SCD channel over a WCD system with a deterministic LD service. If the WCD channel is b_{WCD}-time-bounded and the LD service is r_{LD}-round-bounded, then the SCD channel implementation is b-time-bounded, where $b(r) = b_{WCD}(r_{LD} + 2r)$.*

Proof Sketch. We show that (1) if no process sends a message in a round (say) r_s, then all the processes receive $\perp$ in round r_s; (2) if exactly one process sends a message (say) m in round r_s, then all the processes receive m in the round r_s; and (3) if multiple processes send messages in round r_s, then all the processes receive $\top$ in round r_s.

Properties (1) and (2) are easy to verify based on the following observations. When $n = 1$, at the lone process i, the event $alone(true, r_{LD})_i$ occurs, and when $n > 1$, at each process i, the event $alone(false, r_{LD})_i$ occurs. For property (3) let T denote the set of transmitting processes in round r_s. If $|T| \geq 2$, then there is a message collision. From the properties of a WCD channel, we know that for every process $j \in T$, m'_j is m_j, and for every process $i \notin T$, m'_i is $\top$ (and m_i is $\perp$). Therefore, in the Ack round, for each process i event $sendWCD(\perp, r_{LD} + 2r_s)$ occurs, and consequently, $p2msg_i$ is $\perp$. Given that $a_{LD} = false$, and $m'_i \neq \perp$ at every process i, each process i executes $receive(\top, r_s)_i$. That is, if $|T| \geq 2$, then all processes receive $\top$ in round r_s.

Next we determine a time-bound function b for the SCD channel. Let b_{WCD} denote a time-bound function for the underlying WCD channel. From the pseudocode we know that the deterministic LD service outputs the *alone* events in round r_{LD} of the WCD channel, and subsequently between every pair of events $send(*, r)_i$ and $receive(*, r)_i$, there are exactly two $sendWCD$ events of the form $sendWCD(*, r_{LD} + 2r - 1)_i$ and $sendWCD(*, r_{LD} + 2r)_i$. Therefore, $b(1) = b_{WCD}(r_{LD} + 2)$ and for each round r, $b(r) = b_{WCD}(r_{LD} + 2r)$. □

Additionally, if the underlying LD service is a $\rho_{LD}(\epsilon)$-round-bounded probabilistic LD service, then Algorithm 1 implements a $\rho(r, \epsilon)$-time-bounded probabilistic SCD channel where $\rho(r, \epsilon) = b_{WCD}(\rho_{LD}(\epsilon) + 2r)$. Thus, we have the following result whose correctness proof is similar to Theorem 1.

Theorem 2. *Algorithm 1 implements probabilistic SCD channel over a WCD channel and a probabilistic LD service. If the WCD channel is b_{WCD}-time-bounded and the probabilistic LD service is $\rho_{LD}(\epsilon)$-round-bounded, then the probabilistic SCD channel implementation is $\rho(r, \epsilon)$-time-bounded where $\rho(r, \epsilon) = b_{WCD}(\rho_{LD}(\epsilon) + 2r)$.*

Remark 1. Implementing LD on top of a WCD system augmented with a solution to LE takes just one additional round. First, all the processes elect a leader with the assumed solution to LE. In the next round of the assumed WCD system, (1) every process that is not the leader transmits, outputs $false$, and then halts; (2) concurrently, the leader outputs $true$ iff it receives $\perp$ at the end of this round, and outputs $false$ otherwise. Correctness is straightforward.

5 Algorithms and Lower Bounds for Loneliness Detection

Here, we explore algorithms and lower bounds for Loneliness Detection. Since LD can be solved in SCD systems in a single round, we focus on WCD systems.

5.1 Upper Bounds for LD in WCD Systems

We present two algorithms, one deterministic and the other randomized. The deterministic algorithm solves the deterministic LD problem in $\mathcal{O}(\log(\frac{u}{n-1}))$ rounds whereas the randomized algorithm solves the probabilistic LD problem in $\mathcal{O}(\frac{\log(1/\epsilon)}{n-1})$ rounds with probability $1-\epsilon$, for $\epsilon \in (0,1]$. We also combine these algorithms to solve probabilistic LD with probability 1.

Bitwise Separation Protocol (BSP). BSP solves the deterministic LD problem in WCD systems in $\mathcal{O}(\log \frac{u}{n-1})$ rounds. The algorithm is as follows. Let the ID of each process i be represented as a sequence of bits denoted id_i; since the ID space is of size u, the sequence is $\lceil \log(u) \rceil$ bits long. Starting from the least significant bit, number the bits from 1 to $\lceil \log(u) \rceil$, and let $id_i[k]$ denote the k-th bit of process i's ID. Let $T_k = \{i \in J : id_i[k] = 1\}$[1]. The algorithm proceeds in phases, each phase consisting of a Transmit round and an Ack round.

In the Transmit round of the k-th phase, exactly the processes in T_k that have not yet halted transmit a message. In the Ack round, if a process $i \in J \setminus T_k$ that has not halted receives either a message or $\top$ in the Transmit round, then i sends an "*ack*" message; furthermore, processes in T_k do not send a message in the Ack round. If a process $i \in J$ that has not yet halted either sends or receives an "*ack*" message or receives $\top$ in the Ack round of a given phase k, then i outputs $alone(false, 2k)_i$ and halts.

If $\perp$ is received in all the Ack rounds, then the algorithm terminates at the end of $2\lceil \log u \rceil$ rounds and the lone process (say) j outputs $alone(true, 2\lceil \log u \rceil)_j$. The pseudocode is shown in Algorithm 2.

Theorem 3. *BSP solves the deterministic LD problem and for each process i; some $alone_i$ event occurs after $2\lceil \log(u) \rceil$ rounds if $n = 1$ and within $2(\lceil \log u \rceil - \lceil \log n \rceil + 1)$ rounds if $n > 1$.*

Proof Sketch. From the pseudocode in Algorithm 2, note that every process i performs exactly one $alone_i$ event. First, assume that $alone(true, r)_i$ occurs. From the pseudocode, we see that $r = 2\lceil \log u \rceil$. For the purpose of contradiction, we assume that $n > 1$. Since no $alone(false, *)_i$ event occurs, i never sends an "*ack*" and i never receives an "*ack*" or $\top$ in any Ack round. This can happen only if in the Transmit round of every phase $k \leq \lceil \log u \rceil$, either all the processes transmit or all the processes listen; this implies that all the processes share the same ID. This contradicts our assumption that process IDs are unique. Hence, when $n = 1$, event $alone(true, 2\lceil \log u \rceil)_i$ occurs at the lone process i.

[1] Recall that J is the set of processes comprising the system.

Algorithm 2 Bitwise Separation Protocol

```
Let $m \in \mathcal{M}$ denote a message that $i$ sends to signal its presence in the system.
Process $i$ executes the following:
    for $k := 1$ to $\lceil \log(u) \rceil$
        **Transmit round:**
            if $(id[k] = 1)$ then perform $send(m, 2k-1)_i$
            else perform $send(\perp, 2k-1)_i$
            wait until $receive(m', 2k-1)_i$
        **Ack round:**
            if $(id[k] \neq 1$ and $m' \neq \perp)$ then perform $send(\text{"}ack\text{"}, 2k)_i$
            else perform $send(\perp, 2k)_i$
            wait until $receive(m'', 2k)_i$
            if $(m'' \neq \perp)$ then perform $alone(false, 2k)_i$; **halt.**
    endfor
    perform $alone(true, 2\lceil \log(u) \rceil)_i$
```

Alternatively, assume that $alone(false, r)_i$ occurs. From the pseudocode, we know that $m_i^{''} \neq \perp$ in round r; that is, i either sent or received an "*ack*" message or received $\top$ in the Ack round of phase $r/2$. Therefore, there is at least one other process in the system. Furthermore, since i either sent or received an "*ack*" message or received $\top$ in round r, then the properties of the WCD channel imply that the same is true for all processes. Hence, for each process j, the event $alone(false, r)_j$ occurs.

Now we provide an upper bound on r. Since representing unique IDs among n processes requires $\lceil \log n \rceil$ bits, and since each process ID is of length $\lceil \log u \rceil$ bits, we infer that within the first $\lceil \log u \rceil - \lceil \log n \rceil + 1$ bits of the process IDs, at some bit position k, there exist at least two processes i and j such that $id_i[k] = 1$ and $id_j[k] = 0$. Therefore, in phase k, i transmits and j listens in the Transmit round. Hence, $alone(false, 2k)$ event occurs at each process. $\square$

Random Separation Protocol (RSP). RSP is used to solve probabilistic LD in WCD systems. RSP verifies that $n > 1$ in $2\frac{log(1/\epsilon)}{n-1}$ rounds with probability at least $1 - \epsilon$ where $\epsilon \in (0, 1]$. However, if $n = 1$, RSP does not terminate.

The protocol is identical to BSP except that IDs in RSP are infinite-bit strings in which the bits is chosen independently and uniformly at random. In each phase k, if $id_i[k] = 1$, then i transmits in the Transmit round; otherwise i listens in the Transmit round. It can be verified easily that RSP terminates in the first phase k in which for some pair of processes i and j, $id_i[k] \neq id_j[k]$.

The probability that the k-th bit of the IDs of all processes are identical is $2 \cdot (\frac{1}{2})^n = (\frac{1}{2})^{n-1}$. It follows that, for a given $\epsilon \in (0, 1]$, the probability that RSP does not terminate in $\frac{log(1/\epsilon)}{n-1}$ phases is ϵ.

Theorem 4. *In RSP, if $n > 1$ and $\epsilon \in (0, 1]$, then for every process i the event $alone(false, r)_i$ occurs within the first $\frac{log(1/\epsilon)}{n-1}$ phases, that is, $r \leq 2\frac{log(1/\epsilon)}{n-1}$, with probability at least $1 - \epsilon$.*

Remark 2. For $\epsilon = 2^{-n}$, Theorem 4 implies that, if $n > 1$, then for every process i, the event $alone(false, r)_i$ occurs within the first $2\frac{n}{n-1}$ rounds with probability at least $1 - 2^{-n}$. Thus, for $n > 1$, the number of rounds within which RSP terminates with high probability is always at most 4.

Combined Separation Protocol (CSP). Even though RSP terminates in fewer rounds than BSP with high probability if $n > 1$, it fails to terminate if $n = 1$. We overcome this problem by interleaving RSP and BSP, executing BSP in the odd rounds and RSP in the even rounds; CSP terminates when either BSP or RSP terminates.

Theorem 5. *CSP solves probabilistic LD on WCD systems where*

$$\rho_{LD}(\epsilon) = \begin{cases} 4\lceil \log u \rceil & \text{if } n = 1, \\ 4(\lceil \log u \rceil - \lceil \log n \rceil + 1) & \text{if } \epsilon = 0 \text{ and } n > 1, \\ 4\min\left((\lceil \log u \rceil - \lceil \log n \rceil + 1), \frac{\log(1/\epsilon)}{n-1}\right) & \text{if } \epsilon \in (0,1] \text{ and } n > 1. \end{cases}$$

5.2 Lower Bounds for LD in WCD Systems

In this section, we present lower bounds for both probabilistic and deterministic LD problems in WCD systems.

Lemma 1. *For any LD algorithm A for WCD systems, any $n > 1$, and any round number $r \leq \lceil \log(u) \rceil - \lfloor \log(n-1) \rfloor - 2$, there exists a set J_{LB} of n processes, such that, when A is run on the system consisting of all processes in J_{LB}, the probability that A does not terminate within r rounds, is at least $(\frac{1}{2})^{rn}$.*

Proof Sketch. For each process i, we consider executing A on system S_i consisting of only process i. Without loss of generality, assume that each execution of S_i takes at least r rounds. Consider the probability space of all executions corresponding to r rounds of S_i. For each such execution and each round z, $1 \leq z \leq r$, define $trans_i(z)$ to be $true$ if i transmits in round z, and $false$ otherwise. Denote the probability of events in this space by Pr_i. We define the boolean function dtd_i (*dominating transmission decision*) on $\{1, ..., r\}$ recursively. $dtd_i(1)$ is assigned the value that is more likely to be taken by $trans_i(1)$, *i.e.*, $dtd_i(1) = true$ iff $Pr_i\{trans_i(1) = true\} \geq \frac{1}{2}$. Let $DTD_{i,1}$ denote the event in the probability space Pr_i that $trans_i(1) = dtd_i(1)$. For each $z \geq 2$, we define $dtd_i(z)$ to be the value that is more likely to be taken by $trans_i(z)$, conditioned on $DTD_{i,z-1}$, *i.e.*, $dtd_i(z) = true$ iff $Pr_i\{trans_i(z) = true | DTD_{i,z-1}\} \geq \frac{1}{2}$. We denote by $DTD_{i,z}$ the event that for each round r', $1 \leq r' \leq z$, $trans_i(r') = dtd_i(r')$.

Since for each process i and each round z, $1 \leq z \leq r$, there are two possible values for $dtd_i(z)$, there are 2^r possible values for dtd_i sequences. Since $2^r < \frac{u}{n-1}$, by the Pigeonhole principle, there exists a set J_{LB} of n processes that have identical sequences of dominating transmission decisions, *i.e.*, $\forall i, j \in J_{LB}, dtd_i = dtd_j$. Let S be the system consisting of processes in J_{LB}. For each z, $1 \leq z \leq r$, let $cdtd(z)$ denote the common dominating transmission decision of the processes of J_{LB} in round z.

Consider an execution α in system S. If for each process $i \in J_{LB}$ and each round $z \leq r$ of α, $trans_i(z) = cdtd(z)$, then for each $i \in J_{LB}$ there exists an execution β in S_i such that i cannot distinguish α from β in the first r rounds. However, in α, the only valid output is $alone(false, *)_*$ whereas in β, the only valid output is $alone(true, *)_*$. Hence, j cannot terminate within r rounds. By induction we show that the probability that for each $i \in J_{LB}$ and z, $1 \leq z \leq r$, $trans_i(z) = cdtd(z)$ is at least $(\frac{1}{2})^{rn}$. Hence, with probability at least $(\frac{1}{2})^{rn}$, A does not terminate within r rounds. □

Lemma 2. *For $n = 1$, no LD algorithm for WCD systems guarantees termination within $\lceil \log(u) \rceil - 2$ rounds.*

Theorem 6. *For any LD algorithm A for WCD systems, and any $n > 1$, there exists a set of processes, J_{LB}, where $|J_{LB}| = n$, such that the probability that A terminates within $\min\left(\frac{\log(\frac{1}{\epsilon})}{n}, \lceil \log \frac{u}{n-1} \rceil - 2\right)$ rounds, when run on the system consisting of all processes of J_{LB}, is at most $1 - \epsilon$. For $n = 1$, no LD algorithm for WCD systems guarantees termination within $\lceil \log(u) \rceil - 2$ rounds.*

Proof. For $n > 1$, the proof follows by substituting $r = \min\left(\frac{\log(\frac{1}{\epsilon})}{n}, \lceil \log \frac{u}{n-1} \rceil - 2\right)$ in Lemma 1. For $n = 1$, the proof follows from Lemma 2. □

Theorem 7. *For $n > 1$, no deterministic LD algorithm for WCD systems guarantees termination within $\lceil \log \frac{u}{n-1} \rceil$ rounds. For $n = 1$, no deterministic LD algorithm guarantees termination within $\lceil \log(u) \rceil - 2$ rounds.*

5.3 Revisiting SCD on WCD Systems

In Sect. 4, we presented an implementation of an SCD channel over a WCD channel using an LD service. In Sect. 5.1, we presented the BSP and CSP algorithms that solve deterministic and probabilistic LD, respectively, over WCD. Note that, BSP and CSP do not send any messages on the WCD channel after they terminate. Hence, Algorithm 1 may use the WCD channel after round r in isolation. Therefore, BSP and CSP can be used as an LD service in Algorithm 1 to implement deterministic and probabilistic SCD systems, respectively.

6 Algorithms and Lower Bounds for Leader Election

In this section, we study deterministic and probabilistic LE problems in both SCD and WCD systems and demonstrate matching upper and lower bounds.

6.1 Upper Bounds for LE in SCD Systems

In this section, we present two algorithms: Bitwise Leader Election Protocol (BLEP) and Combined Leader Election Protocol (CLEP). The former is a deterministic algorithm which solves deterministic LE in SCD systems. The latter is a randomized algorithm which interleaves BLEP and Nakano and Olariu's algorithm in [9] to solve probabilistic LE in SCD systems.

Algorithm 3 Bitwise Leader Election Protocol

```
$active = true$
for $r := 1$ to $\lfloor \log(u) \rfloor + 1$
    if ($active = true$ and $id_i[r] = 1$) then perform $send(id_i, r)_i$
    else perform $send(\perp, r-1)_i$
    wait until $receive(m', r-1)_i$
    if ($m' \in I$) then perform $leader(m')_i$; **halt.**
    if ($m' = \top$) then $active = (active) \& (id_i[r] = 1)$
endfor
```

Bitwise Leader Election Protocol (BLEP). BLEP solves deterministic LE in SCD systems in $\mathcal{O}(\log u)$ rounds. In BLEP, every process contending to be the leader is *active*, and *inactive* otherwise; a Boolean variable *active* denotes whether or not a process is active. Initially, all the processes are *active*. The execution proceeds from round 1 to round $\lfloor \log u \rfloor + 1$. In each round r, every process i such that i is *active* and $id_i[r] = 1$, transmits its ID id_i; all other processes are silent in round r. At the end of round r, every process j receives some response from the SCD channel. If j receives a collision notification $\top$ at the end of round r, and j was *active* but did not transmit its ID in round r (because $id_j[r] = 0$), then j ceases to be *active* (becomes inactive) at the end of round r and therefore stops contending to be the leader. On the other hand, if the response at the end of round r is the ID of some process l, then j elects l as the leader, outputs $leader(l)_j$, and halts. If j receives $\perp$ at the end of round r, then j does nothing. The execution proceeds to round $r+1$, and so on, until round $\lfloor \log u \rfloor + 1$. The pseudocode is shown in Algorithm 3.

Theorem 8. *BLEP solves the LE problem within* $\lfloor \log(u) \rfloor + 1$ *rounds.*

Proof Sketch. Establishing the safety properties of LE is straightforward and follows from the pseudocode. Next, we prove that some *leader* event occurs within $\lfloor \log u \rfloor + 1$ rounds.

From the pseudocode, we see that if i and j are active in round r, then for each r', $1 \leq r' \leq r$, $id_i[r'] = id_j[r']$. Therefore, at the end of $\lceil \log u \rceil$ rounds, there can be at most two active processes in the system. If just one process (say) i remains active, then consider the earliest round $r \leq \lfloor \log u \rfloor$ at the end of which i is the only active process. By construction, multiple processes are active at the beginning of round r. Since i is the only process active at the end of round r, for each process $j \neq i$ that is active in round r, $id_j[r] = 0$ and $id_i[r] = 1$. Hence, in round r, only i transmits its ID, and all the processes in the system receive i's ID. Therefore, for each process j in the system, $leader_j$ event occurs in round $r \leq \lfloor \log u \rfloor$ and contradicts our assumption that no *leader* event occurs by the end of round $\lfloor \log u \rfloor$.

On the other hand, if two processes (say) i and j remain active, then the first $\lfloor \log u \rfloor$ bits of their IDs are identical. Therefore, the last bit of their IDs must be different. Without loss of generality, let $id_i[\lfloor \log u \rfloor + 1] = 1$ and $id_j[\lfloor \log u \rfloor + 1] =$

0. In round $\lfloor \log u \rfloor + 1$ only i transmits its ID and therefore, for each process i, $leader_i$ event occurs in round $\lfloor \log u \rfloor + 1$. ☐

Combined Leader Election Protocol (CLEP). Nakano and Olariu [9] present a randomized LE algorithm for SCD systems that terminates in $\mathcal{O}(1)$ rounds if $n = 1$, and in $\mathcal{O}(\log\log n + \log(\frac{1}{\epsilon}))$ rounds with probability at least $1 - \epsilon$ if $n > 1$. CLEP interleaves Nakano-Olariu's algorithm with BLEP, by executing BLEP in the odd rounds and Nakano-Olariu in the even rounds. The time complexity of CLEP matches the lower bound for probabilistic LE.

Theorem 9. *CLEP solves the LE problem in SCD systems and terminates within $\rho_{LE}(\epsilon)$ rounds with probability at least $1 - \epsilon$ where:*

$$\rho_{LE}(\epsilon) = \begin{cases} \mathcal{O}(1) & \text{if } n = 1, \\ \mathcal{O}(\log u) & \text{if } \epsilon = 0 \text{ and } n > 1, \\ \mathcal{O}(\min(\log u, \log\log n + \log(\frac{1}{\epsilon})) & \text{if } \epsilon \in (0,1] \text{ and } n > 1. \end{cases}$$

6.2 Lower bounds for LE in both SCD and WCD Systems

Here we present lower bounds for deterministic and probabilistic LE in SCD systems in Theorems 10 and 12, respectively, and they match the upper bounds presented in Theorems 8 and 9, respectively. Note that these lower bounds hold for the weaker WCD systems as well. We demonstrate these lower bounds by proving the lower bounds for η-LE, and since η-LE is equivalent the LE problem where $n = \eta$ is known, lower bounds for η-LE hold for LE as well.

Lemma 3. *For any $\eta > 1$ and any randomized η-LE algorithm A in SCD systems, there exist some set J of η processes such that there is a non-zero probability that A, when run on J, does not terminate within $\lceil \log \frac{u}{\eta - 1} \rceil - 2$ rounds.*

Proof Sketch. Let $k = \lceil \log \frac{u}{\eta - 1} \rceil - 2$. Assume for contradiction that there exists an $\eta > 1$ and an η-LE algorithm A that terminates within k rounds with probability 1. We construct executions of A for each process i in which i receives $\top$ from the channel in each round that it transmitted and receives $\perp$ otherwise. Using techniques from Lemma 1, we show that there exists an execution of A on some set J_{LB} of η processes for which either each process $i \in J_{LB}$ elects itself as the leader or no process is elected leader within k rounds. This violates the properties of η-LE and forces the contradiction. ☐

Theorem 10. *For any $n > 1$, no deterministic LE algorithm in SCD systems can guarantee termination within $\lceil \log \frac{u}{n-1} \rceil - 2$ rounds.*

The proof follows from Lemma 3. Next we derive a lower bound for termination probability of 2-LE and extend it to the LE problem.

Lemma 4. *Let A be any 2-LE algorithm in SCD systems and suppose that $r < \lceil \log(u) \rceil - 1$ and $2 \leq u$. There exist two processes such that the probability of termination of A within k rounds, when A is run on the system of those two processes, is at most $1 - (\frac{1}{4})^{r+1}$.*

Proof Sketch. The proof structure is similar to that of Lemma 1. The key difference is the following. In the proof of Lemma 1 we considered some specific executions of an LD algorithm A_{LD} in WCD systems with just one process and showed that such executions are locally indistinguishable from some (other) specific executions of A_{LD} in a WCD system with a specific set of n processes. In SCD systems, such a construction is not feasible for the following reason. In WCD systems when a transmitting process always receives the same feedback from the channel. On the other hand, in SCD systems, a transmitting process could receive different feedback depending on whether or not a collision occurred. To circumvent this issue, we consider executions of A, a solution to 2-LE problem in SCD systems, in a fake scenario where a process receives $\top$ in every round that it transmits. We use such executions to demonstrate that with probability at least $(\frac{1}{4})^{r+1}$, A does not terminate. $\square$

Theorem 11. *For any $u \geq 2$, any ID space I of size u, any $\epsilon \in (0,1]$, and any 2-LE algorithm A in SCD systems, there exist two processes with IDs from I such that, when A is run with just those two processes, the probability that A terminates within $\min(\log(\frac{1}{4\epsilon})/2, \lceil\log(u)\rceil - 2)$ rounds is at most $1-\epsilon$.*

Theorem 12. *For any $u \geq 2$, any ID space I of size u, any $\epsilon \in (0,1]$, any η, $1 \leq \eta \leq \frac{u}{2}$, and any 2η-LE algorithm A in SCD systems, there exist 2η processes with IDs from I such that, when A is run with just those 2η processes, the probability that A terminates within $\min(\log(\frac{1}{4\epsilon})/2, \lceil\log(\frac{u}{n})\rceil - 2)$ rounds is at most $1-\epsilon$.*

Proof Sketch. Assume for the sake of contradiction that there exists some 2η-LE algorithm A that terminates within $\min(\log(\frac{1}{4\epsilon})/2, \lceil\log(\frac{u}{n})\rceil - 2)$ rounds with probability greater than $1-\epsilon$. Consider an ID space I^* of size $u^* = \lfloor\frac{u}{\eta}\rfloor$. Using A we construct a 2-LE algorithm A^* that emulates A on groups of processes and terminates when A does. Since A terminates within $\min(\log(\frac{1}{4\epsilon})/2, \lceil\log u^*\rceil - 2)$ rounds with probability greater than $1-\epsilon$, the same bounds apply to A^* as well, and this contradicts contradicts Theorem 11. $\square$

6.3 Leader Election in Weak Collision Detection Systems

In this section, we show that the LE problem in WCD systems can be solved in time complexities that match the lower bounds presented in Sect. 6.2 for both deterministic and probabilistic cases. We can solve LE on WCD systems by first implementing SCD systems on WCD systems as presented in Sect. 5.3, and then solving LE on the thus constructed SCD systems using BLEP and CLEP from Sect. 6.1. Thus, we have the following results.

Theorem 13. *For a WCD system with IDs from an ID-space of size u and consisting of n processes, $1 \leq n \leq u$, there exists a deterministic LE algorithm with a time-bound function given by $r_{LE} = \mathcal{O}(\log u)$ rounds.*

Note that the time complexity above, $\mathcal{O}(\log u)$, matches the $\Omega(\log\frac{u}{n})$ lower bound presented in Lemma 3 asymptotically when $n << u$.

Theorem 14. *For a WCD system with IDs from an ID-space of size u and consisting of n processes, $1 \leq n \leq u$, there exists a randomized LE algorithm with a time-bound function $\rho_{LE}(\epsilon)$ where for any $\epsilon \in (0, 1]$,*

$$\rho_{LE}(\epsilon) = \begin{cases} b_{WCD}(\mathcal{O}(\log(u))) & \text{if } n = 1 \\ b_{WCD}(\mathcal{O}(\min(\log u, \log\log n + \log(\frac{1}{\epsilon}))) & \text{if } n > 1. \end{cases}$$

The upper bounds presented above match the respective lower bounds. For $n = 1$, Theorem 6, along with the reduction of LD to LE in the Remark 1, shows an $\Omega(\log u)$ lower bound for LE in WCD systems which matches the upper bound. For $n > 1$, the upper bound presented above matches the lower bound presented in Theorem 12, when $\epsilon = \mathcal{O}(\frac{1}{n})$.

References

1. Bordim, J.L., Ito, Y., Nakano, K.: Randomized leader election protocols in noisy radio networks with a single transceiver. In: Proceedings of the 4th International Symposium on Parallel and Distributed Processing and Applications. pp. 246–256 (2006), http://dx.doi.org/10.1007/11946441_26
2. Capetanakis, J.I.: Tree algorithms for packet broadcast channels. IEEE transactions on information theory 25(5), 505–515 (1979), http://dx.doi.org/10.1109/TIT.1979.1056093
3. Clementi, A.E.F., Monti, A., Silvestri, R.: Distributed broadcast in radio networks with unknown topology. Theoretical Computer Science 302, 337–364 (2003), http://dx.doi.org/10.1016/S0304-3975(02)00851-4
4. Ghaffari, M., Lynch, N., Sastry, S.: Leader election using loneliness detection. Tech. Rep. MIT-CSAIL-TR-2011-xxx, CSAIL, MIT (2011)
5. Hayes, J.: An adaptive technique for local distribution. IEEE transactions on communication 26, 1178–1186 (1978), http://dx.doi.org/10.1109/TCOM.1978.1094204
6. Kowalski, D., Pelc, A.: Broadcasting in undirected ad hoc radio networks. Distributed Computing 18(1) (2005), http://dx.doi.org/10.1007/s00446-005-0216-7
7. Kowalski, D., Pelc, A.: Leader election in ad hoc radio networks: A keen ear helps. In: International Conference on Automata, Languages and Programming. pp. 521–533 (2009), http://dx.doi.org/10.1007/978-3-642-02930-1_43
8. Nakano, K., Olariu, S.: Randomized leader election protocols in radio networks with no collision detection. In: Proceedings of the 11th International Conference of Algorithms and Computation. pp. 362–373 (2000), http://dx.doi.org/10.1007/3-540-40996-3_31
9. Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE transactions on parallel and distributed systems 13(5) (2002), http://dx.doi.org/10.1109/TPDS.2002.1003864
10. Schneider, J., Wattenhofer, R.: What is the use of collision detection (in wireless networks)? In: Proceedings of the International Symposium on Distributed Computing. pp. 133–147 (2010), http://dx.doi.org/10.1007/978-3-642-15763-9_14
11. Willard, D.E.: Log-logarithmic selection resolution protocols in a multiple access channel. SIAM Journal of Computing 15, 468–477 (1986), http://dx.doi.org/10.1137/0215032