Robust and Optimal Contention Resolution without Collision Detection*

Yonggang Jiang
Max Planck Institute for Informatics
Saarland Informatics Campus
Germany
yjiang@mpi-inf.mpg.de

Chaodong Zheng
State Key Laboratory for Novel Software Technology
Nanjing University
China
chaodong@nju.edu.cn

ABSTRACT

Contention resolution on a multiple-access communication channel is a classical problem in distributed and parallel computing. In this problem, a set of nodes arrive over time, each with a message it intends to send. Time proceeds in synchronous slots, and in each slot each node can broadcast its message or remain idle. If in a slot one node broadcasts alone, it succeeds; otherwise, if multiple nodes broadcast simultaneously, messages collide and none succeeds. Nodes can differentiate collision and silence (that is, no node broadcasts) only if a collision detection mechanism is available. Ideally, a contention resolution algorithm should satisfy at least three criteria: (a) low time complexity (i.e., high throughput), meaning it does not take too long for all nodes to succeed; (b) low energy complexity, meaning each node does not make too many broadcast attempts before it succeeds; and (c) strong robustness, meaning the algorithm can maintain good performance even if interference is present. Such interference is often modeled by jamming—a jammed slot always generates collision.

Previous work has shown, with collision detection, there are "perfect" contention resolution algorithms satisfying all three criteria. On the other hand, without collision detection, it was not until 2020 that an algorithm was discovered which can achieve optimal time complexity and low energy cost, assuming there is no jamming. More recently, the trade-off between throughput and robustness was studied. However, an intriguing and important question remains unknown: without collision detection, are there "perfect" contention resolution algorithms? In other words, when collision detection is absent and jamming is present, can we achieve both low total time complexity and low per-node energy cost?

In this paper, we answer the above question affirmatively. Specifically, a new randomized algorithm for robust contention resolution is developed, assuming collision detection is not available. Lower bound results demonstrate it achieves both optimal time complexity and optimal energy complexity. If all nodes start execution simultaneously—which is often referred to as the "static case" in literature—another algorithm is developed that runs even faster. The separation on time complexity suggests, for robust contention resolution without collision detection, "batch" instances (that is, nodes start simultaneously) are inherently easier than "scattered" ones (that is, nodes arrive over time).

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

Contention resolution; throughput; energy; jamming; backoff.

ACM Reference Format:
Yonggang Jiang and Chaodong Zheng. 2022. Robust and Optimal Contention Resolution without Collision Detection. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '22), July 11–14, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3490148.3538592

1 INTRODUCTION

In computer systems, there are many scenarios in which a collection of players contend to access a shared resource. For example, a set of processes try to access a shared file on a hard drive or a table in a database, a set of radio transceivers each with a packet try to send these packets over a wireless channel, a set of users each with a document try to print them out using a printer, etc. In these settings, usually the goal is to let each player successfully access the shared resource (at least) once. However, the challenge lies in the requirement that successful accesses must be *mutual exclusive*: if two or more players make access attempts simultaneously, a collision occurs and all these attempts fail.

In distributed and parallel computing, the above problem is known as *contention resolution*. In studying this problem, often the shared resource is modeled as a synchronous multiple-access communication channel, and each player is modeled as a node with a message that needs to be sent over this channel. (See, e.g., [6, 7, 11, 13, 15, 22].) More specifically, the system proceeds in synchronous time slots. Each player joins the system at the beginning of some slot, but players do not have access to any global clock. In each slot, each player may try to send its message by broadcasting it on the shared communication channel, or remain idle. If in a slot only one player u broadcasts, it succeeds in that slot and all players are informed of this success. Moreover, by the end of that slot, u will halt and exit the system. On the other hand, if in a slot multiple players broadcast simultaneously, then all of them fail as messages collide with each other. In such a case, the exact channel feedback depends on the availability of a *collision detection* mechanism. In particular, when collision detection is available, for each slot not containing a success, the channel will inform all players whether the slot is silent (that is, no node broadcasts) or noisy (that is, mul-

*The full version of this paper is available at https://arxiv.org/abs/2111.06650.

SPAA '22, July 11–14, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9146-7/22/07.
https://doi.org/10.1145/3490148.3538592

tiple nodes broadcast and a collision occurs). By contrast, without collision detection, the channel feedback for each slot is binary: either the slot contains a success, or not.

Performance metrics. To evaluate the performance of contention resolution algorithms, often the following three metrics are used:

Throughput (Time complexity). Achieving a high throughput is usually the primary objective of any contention resolution algorithm. Although the exact definition of throughput depends on the specific model being considered, intuitively it captures the number of messages the algorithm can successfully transmit within a given period of time, representing how efficient can the algorithm process incoming requests. In synchronous systems, since each slot contains at most one successful transmission, constant throughput is the best result any contention resolution algorithm can hope for; that is, generating n successes within $\Theta(n)$ slots.

Number of access attempts (Energy complexity). Due to collisions, it is likely that each player has to make multiple access attempts before succeeding. However, in many scenarios, there is a certain cost associated with each attempt. A prominent example is radio networks: modern lightweight wireless devices are often battery powered, and the energy cost of emitting or receiving radio signals might be large when compared with computation [25]. Therefore, contention resolution algorithms also try to minimize participating players' number of access attempts. In studying contention resolution, especially in the context of radio networks, researches often call the maximum number of access attempts any node may make as the *energy complexity* of the algorithm. (See, e.g., [6, 9, 10, 15, 22].) In this paper, we also adopt this terminology.

Robustness. Collision is not the only possible reason a transmission attempt fails. Participating nodes, or sometimes even the communication channel itself (e.g., when the channel models a shared printer), could suffer hardware or software errors. Even without internal failures, external interference might exist. For example, a wireless link may be affected by electromagnetic noise, a server may be the victim of denial-of-service attacks, etc. In studying contention resolution, these interference are often modeled by *jamming*. (See, e.g., [3, 6, 11, 13].) Formally, if a slot is jammed, a collision occurs in that slot, regardless of the actual number of broadcasting nodes. Robust contention resolution algorithms are preferable, sometimes even necessary. However, as we detail below, jamming could affect the complexity of the problem.

Existing results. Given its importance and wide applicability, it is not surprising that contention resolution is extensively studied. Nonetheless, many important breakthroughs are only made recently. In particular, it was not until 2018 that a "perfect" contention resolution algorithm was proposed which simultaneously achieves high throughput and low energy complexity, even when adversarial jamming is present [6]. In particular, assuming n nodes are injected over time by an adversary, and d slots could be jammed by the adversary, Bender, Fineman, Gilbert, and Young proposed an algorithm called Re-Backoff guaranteeing: (a) constant throughput in expectation; and (b) each node makes $O(\log^2(n+d))$ access attempts in expectation. In 2019, Chang, Jin, and Pettie [11] proposed another algorithm that can achieve similar guarantees, by using a multiplicative weight update scheme. Compared with [6], this new algorithm is simpler and achieves higher throughput (though the asymptotic throughput of these two algorithms are identical).

Both of the above two algorithms rely on the availability of collision detection. Generally speaking, if a node observes a noisy slot or many collisions within an interval, then the node can infer that the channel is congested and it should "backoff" by not sending its message. By contrast, if a node observes a silent slot or many empty slots within an interval, then the node should "backon" and send more frequently. Fundamentally, collision detection explicitly reveals why a slot fails, thus allowing nodes to act accordingly in the following slots. As a result, without collision detection, contention resolution becomes harder.

In real world, although in many cases detecting collision is feasible (e.g., in Ethernet, "carrier-sense multiple access with collision detection" (CSMA/CD) is part of the MAC protocol [21]), there also exist cases in which detecting collision becomes more challenging or complicates software implementation. For instance, in wireless networks, a standard radio transceiver often cannot simultaneously listen and transmit, thus it becomes harder to perform "carrier sensing" to detect collision. As a result, in the literature, both the scenarios of with and without collision detection are considered practical and studied.

Unfortunately, for a very long time, it was unclear whether achieving constant throughput for contention resolution is possible if collision detection is not available, even without any interference.[1] Before 2020, the best result was from De Marco and Stachowiak [15]: if nodes arrive over time and no errors occur, then each injected node succeeds within $O((n\ln^2 n)/\ln\ln n)$ slots.[2]

Two years ago, Bender, Kopelowitz, Kuszmaul, and Pettie [7] resolved the aforementioned long standing problem by providing an algorithm that achieves constant throughput without collision detection, even when the nodes are injected adversarially. They also proved a lower bound showing constant throughput is impossible when jamming is present. This demonstrates a fundamental separation between contention resolution with and without collision detection. More recently, Chen, Jiang, and Zheng [13] extended [7] by taking adversarial interference into consideration. In particular, for any level of jamming ranging from none to constant fraction, they prove an upper bound on the best possible throughput, along with an algorithm attaining that bound.

However, one important piece is still missing in the big picture. Although the algorithms in [13] are able to achieve optimal throughput for any given level of jamming, their (as well as the algorithms in [7]) energy complexity could be high when jamming is present. In particular, assuming the adversary jams d slots, there are nodes who might have to make $\Omega(\sqrt{d})$ attempts before succeeding. By contrast, when collision detection is available, achieving similar guarantees only requires $O(\text{poly-log}(n+d))$ attempts. Therefore, the intriguing question is: without collision detection, are there "perfect" contention resolution algorithms? In other words, when collision detection is not available and jamming is present, are there contention resolution algorithms that can achieve best possible

[1] Nonetheless, if all nodes are injected simultaneously, then algorithms achieving constant throughput without collision detection were already known for a while. See, e.g., the paper by Mosteiro, Fernandez Anta, and Muñoz [24].

[2] In the same paper, the authors also showed that if nodes are allowed to include control bits in their broadcast contents and may stay in the system after successful transmissions, then constant throughput is possible.

throughout, while maintaining low per-node energy cost?

Contribution and new results. This paper answers above question affirmatively: we design two new randomized algorithms for robust contention resolution, without using collision detection.

Our first algorithm works for the case where n nodes are injected over time by an adversary, which we refer to as the *dynamic case*. The time complexity—that is, the number of slots required to let all nodes succeed—of this algorithm is $O(n \log n + d)$, with high probability in $n+d$. (Throughout the paper, we say an event happens "with high probability (w.h.p.)" in some parameter λ if it happens with probability at least $1-1/\lambda^{\beta}$, for some desirable constant $\beta \geq 1$.) This is optimal given the lower bound proved in [13]. As for energy complexity, our algorithm ensures the number of access attempts any node made is $O(\log^2 n + \log^2 d)$, w.h.p. in $n + d$. Once again, we prove a matching lower bound demonstrating its optimality. Technically, this algorithm draws inspirations from various existing work: readers with expertise in contention resolution will find some design and analysis techniques familiar. Nevertheless, we stress that it is non-trivial to correctly employ and modify these existing toolkits to obtain a desired algorithm. Moreover, deriving a clean and concise analysis for the dynamic case algorithm also becomes more challenging when jamming is present.

Our second algorithm shows if a stronger level of "synchrony" is provided, then constant throughput is attainable. Specifically, if n nodes start execution simultaneously, which we refer to as the *static case*, then all of them can succeed within $O(n + d)$ slots, w.h.p. in $n + d$. Clearly, this is optimal: each slot can generate at most one success, and the adversary can jam d slots to block any communication. This second algorithm also ensures w.h.p. in $n + d$ the energy complexity of each node is $O(\log^2 n + \log^2 d)$, which almost matches the $\Omega(\log \log n + \log^2 d)$ lower bound we proved. Inside this algorithm is a novel two stage approach: the first phase of it mainly handles the scenario where jamming from the adversary is weak, whereas the second phase handles the scenario where jamming is strong. A particularly interesting point is the condition for transitioning from phase one to phase two, as it works for arbitrary d values, without knowing any estimate of d. We believe it could be potentially used in other algorithms to achieve robustness while maintaining efficiency.

Before stating the results formally, we clarify some additional model details and assumptions. We often call the adversary Eve, and she is adaptive. Before execution starts, Eve decides a value n, meaning n nodes will be injected into the system. Eve also has a jamming budget d, meaning she can jam up to d slots. Nodes do not know the value of n or d. In the static case, Eve injects (i.e., activates) all nodes at the beginning of slot one; whereas in the dynamic case, she can inject nodes in an arbitrary fashion. Therefore, in the dynamic case, there might be slots in which there are no active nodes, yet some nodes are still not injected by Eve. (Recall a node halts and leaves once its message is successfully sent.) We say a slot is an *active slot* if in that slot at least one node is active. The adaptivity of Eve is reflected by the assumption that, at the beginning of each slot, Eve is given the past behavior of all nodes, and she can use this information to determine her behavior in the current slot. (Specifically, whether to inject any new nodes and whether to jam this slot.) However, Eve does not know active nodes' behavior in the current slot.

The following definition introduces (f, g)-*time-cost* and (f, g)-*energy-cost*. It formalizes the notation of throughput and energy complexity, and simplifies later presentation.

*Definition 1.1 (**Throughput and Energy Complexity**).* Let $\mathcal{A}$ be the algorithm each node runs after its activation. Let $f, g : \mathbb{N}^+ \to \mathbb{R}^+$ be two functions.

- An *active slot* is a slot in which at least one node is active.
- Algorithm $\mathcal{A}$ achieves (f, g)-*time-cost* if there exists a constant C such that for any integer $n, d \geq 1$ and any adaptive adversary that injects n nodes and has jamming budget d, the total number of active slots is at most $C \cdot (f(n) + g(d))$, with high probability in $n + d$.[3]
- Algorithm $\mathcal{A}$ achieves (f, g)-*energy-cost* if there exists a constant C such that for any integer $n, d \geq 1$ and any adaptive adversary that injects n nodes and has jamming budget d, the max number of broadcasting attempts any node made is at most $C \cdot (f(n) + g(d))$, with high probability in $n + d$.

The following two theorems state our algorithmic results, notice the difference on the time complexity.

Theorem 1.2 (**Dynamic Case Upper Bound**). *There exists an algorithm achieving $(n \log n, d)$-time-cost and $(\log^2 n, \log^2 d)$-energy-cost. That is, this algorithm generates n successes within $O(n \log n + d)$ active slots, and each node makes $O(\log^2 n + \log^2 d)$ access attempts, with high probability in $n + d$.*

Theorem 1.3 (**Static Case Upper Bound**). *If all nodes start simultaneously, then there exists an algorithm achieving (n, d)-time-cost and $(\log^2 n, \log^2 d)$-energy-cost. That is, this algorithm generates n successes within $O(n + d)$ active slots, and each node makes $O(\log^2 n + \log^2 d)$ access attempts, with high probability in $n + d$.*

The following two theorems state our lower bound results.

Theorem 1.4 (**Dynamic Case Lower Bound**). *If an algorithm achieves (f_t, g_t)-time-cost and (f_e, g_e)-energy-cost with $g_t(d) = d$, then $f_t(n) = \Omega(n \log n)$, $f_e(n) = \Omega(\log^2 n)$, and $g_e(d) = \Omega(\log^2 d)$.*

Theorem 1.5 (**Static Case Lower Bound**). *If all nodes start execution simultaneously and an algorithm achieves (f_t, g_t)-time-cost and (f_e, g_e)-energy-cost with $f_t(n) = n$ and $g_t(d) = d$, then $f_e(n) = \Omega(\log \log n)$ and $g_e(d) = \Omega(\log^2 d)$.*

We conclude this part by noting that our lower bounds hold even for a weaker oblivious adversary.

Additional related work. Perhaps the most classical algorithm to resolve contention is *binary exponential backoff*. One standard implementation of it is to let each node send its message with probability $1/i$ in the i-th slot since the node's activation. Despite binary exponential backoff is widely used in practice (e.g., Ethernet and WiFi networks [21], concurrency control in operating systems and database management systems [26, 28]), it is long known that this scheme cannot always achieve optimal throughput [2, 5, 20]. Therefore, many variants are proposed and analyzed, such as quadratic backoff [20], saw-tooth backoff, loglog-iterated backoff [5]. Our

[3] In this paper, we require with high probability in $n + d$ (instead of just n) since we want the failure probability to go down as the time cost grows up (recall time cost depends on both n and d).

algorithms also utilize variants of binary exponential backoff.

As mentioned previously, the availability of collision detection greatly affects the performance of contention resolution algorithms. Around 2010, a series of elegant results were published (see, e.g., [4, 27]), demonstrating how constant throughput could be attained by using collision detection, even if jamming is present, in the context of single-channel wireless networks. In these works, the considered adversary is so-called "rate limited": for any sufficiently long time interval, the adversary can only jam a limited fraction (e.g., a constant fraction) of slots. Over the last few years, similar results were also obtained in the standard multiple-access communication channel model with a fully adaptive jamming adversary [6, 11]. However, it was not until 2020 that an algorithm was developed which can achieve constant throughput without collision detection [7]. This paper also focuses on the more challenging scenario where collision detection is not available.

Besides throughput, number of access attempts before succeeding is another key performance metric. This is especially relevant in radio networks since energy consumption of radio transceivers often dominate the total energy expenditure of wireless devices [25], and contention resolution is closely related to many fundamental communication primitives in radio networks [9, 10, 12, 17]. Therefore, the number of channel accesses is also referred to as energy complexity in the literature. Existing contention resolution algorithms achieving good throughput usually have energy complexity that are poly-logarithmic in the number of participating nodes, though [8] shows in fact $O(\log(\log^*(n)))$ accesses are sufficient in expectation.[4] However, when collision detection is not available and jamming is present, to the best of our knowledge, no existing work can achieve good throughput while maintaining low energy complexity. Our paper addresses this open problem.

Although this paper and many previous work assume worst-case (i.e., adversarial) arrival pattern, another major line of research assumes the arrivals of nodes follow some statistical pattern. In those works (e.g., [2, 18, 19]), often the main objective is to analyze the maximum stable packet arrival rate for various contention resolution algorithms.

It is also worth noting, if a single success is sufficient (instead of requiring every node to succeed once), then contention resolution degrades to another classical symmetry breaking problem: leader election. Leader election is used implicitly in many contention resolution algorithms, [15] is a recent example. A classical result by Willard [29] shows a tight $\Theta(\log\log n)$ bound for leader election, assuming n nodes start simultaneously and collision detection is available. More recent results consider more diverse settings (e.g., [12, 14, 16]). Interestingly, it seems our lower bounds could also apply to the leader election problem, as in deriving them we consider the time and energy required to generate the first success.

Lastly, we stress that contention resolution is an extensively studied problem, only the most relevant results are briefly mentioned here, and many interesting works are not discussed. For example, a recent trend is focusing on deterministic algorithms [3, 22]; there are also papers considering messages with delivery deadlines [1].

[4] In fact, this $O(\log(\log^*(n)))$ bound counts both the number of "send" and "listen", where "listen" means obtaining channel feedback for one slot. In this paper, we only consider "sending complexity" and assume channel feedback is provided for free. This assumption is used in many works studying contention resolution. On the other hand, the assumption that "listen" is not free is usually made in the context of radio networks.

Paper outline. In the next section, we give an overview on the design and analysis of the two new contention resolution algorithms, as well as the key ideas we exploit in proving the lower bounds. Then, in Section 3 and Section 4, we introduce the two algorithms in detail. For each of these two sections, we will first give a more through discussion on the intuition, then present algorithm pseudocode, and finally proceed to the analysis. We will conclude this paper by proving the lower bounds on time complexity and energy complexity. Due to space constraints, omitted proofs are provided in the full version of the paper.

2 TECHNICAL OVERVIEW

Contention. When designing efficient contention resolution algorithms, the key is to maintain a proper *contention* on the communication channel throughout the execution. Specifically, the contention of a slot on a channel is defined to be the sum of the broadcasting probabilities of all active nodes. By definition, the contention of a slot denotes the expected total energy expenditure of nodes in that slot. On the other hand, the contention of a slot also indicates the likelihood of a slot being a successful one. Particularly, the follow lemma holds, where n denotes the number of active nodes in a slot, p_i denotes the broadcast probability of node i, and p denotes the contention of that slot. (See full paper for its proof.)

Lemma 2.1. *Let $n \in \mathbb{N}^+$, let X_i be an indicator random variable with $\Pr[X_i = 1] = p_i$ for all $i \in [n] = \{1, 2, \cdots, n\}$, and let $p = \sum_{i=1}^{n} p_i$. If $X_1, X_2, ..., X_n$ are independent, then:*

- $\Pr\left[\left(\sum_{i=1}^{n} X_i\right) = 1\right] \geq \min\{4^{-p}, p/4\}$ *when all* $p_i \in [0, 1/2]$
- $\Pr\left[\left(\sum_{i=1}^{n} X_i\right) = 0\right] \geq 4^{-p}$ *when all* $p_i \in [0, 1/2]$
- $\Pr\left[\left(\sum_{i=1}^{n} X_i\right) = 1\right] \leq p \cdot e^{-p+1}$

Some important implications of the above lemma are: (a) if the contention of a slot is some constant, then the probability that this slot generates a success is some constant; (b) if the contention of a slot is sufficiently large in $\Omega(\log\lambda)$ where $\lambda > 1$, then with high probability in λ this slot will not generate a success; and (c) if the contention of a slot is sufficiently small in $O(\log\lambda)$ where $\lambda > 1$, then the success probability in this slot is at least $1/\lambda^{\beta}$ for some constant $0 < \beta < 1$. Notice (b) and (c) together imply $\Theta(\log\lambda)$ is the "right" contention if we want to create a success in $\Theta(\lambda)$ slots.

Exponential backoff. Recall a standard way to implement binary exponential backoff is to let each node broadcast with probability $1/i$ in the i-th slot since its arrival. Consider the scenario in which n nodes start running binary exponential backoff simultaneously, observe how the contention evolves. In the first n slot, the contention is $\Omega(1)$ and limited successes will occur; particularly, at least a constant fraction of all n nodes will remain active by the end of slot n. Then, from slot $n+1$ to, say, slot $10n$, the contention will remain to be a constant. By Lemma 2.1, we know in expectation a success will occur every some constant slots. Thus, by the end of slot $10n$, at most some constant fraction of all n nodes will remain active. Lastly, from slot $10n+1$, the contention will continue to drop, and eventually reach $o(1)$, which is too small for successful transmissions to occur frequently. In short, the first $\Theta(n)$ slots of binary exponential backoff are efficient in that $\Theta(n)$ successes are likely to occur, and each node's energy cost is $O(\log n)$. Nonetheless,

beyond this interval, throughput will drop. In this paper, we will use both standard exponential backoff and its variants. To simplify presentation, we define the following generalized backoff pattern.

Definition 2.2 (h-BACKOFF). Let $h : \mathbb{N}^+ \to \mathbb{R}^+$ be a function. We say a node runs h-BACKOFF from slot s, if for any $x \in \mathbb{N}^+$, the node sends its message with probability $\min\{1, h(x)\}$ in slot $s + x - 1$.

For example, $1/x$-BACKOFF is the standard exponential backoff.

The two-channel model. When designing and analyzing our algorithms, it is often more convenient to assume there are *two* channels. Here, we introduce the two-channel model, and defer the discussion on how to covert a two-channel algorithm to a single-channel algorithm to later parts of the paper.

Simply put, the two channels act like two independent multiple-access communication channels: in each slot, for each of the two channels, each active node and the adversary can independently decide its behavior on this channel: broadcast its message, jam the channel, or remain idle. In each slot, both channels give feedback to the active nodes. Time complexity naturally extends to the two-channel model. As for energy complexity, if in a slot a node sends on both channels (respectively, Eve jams both channels), then the energy expenditure of this node (respectively, the jamming budget Eve spends) is counted as two.

Overview of algorithm for the dynamic scenario. Recall we have argued above that the first $\Theta(n)$ slots of binary exponential backoff are efficient. In fact, in previous works that achieve constant throughput without using collision detection, a core idea is to "repeat the efficient part of the exponential backoff process" [7, 13]. However, when jamming is present, the energy complexity of this scheme grows quickly. Specifically, for each repetition of exponential backoff, Eve can jam all the first $\Theta(n)$ slots and then allow one success to occur so that the next repetition starts. If this process is repeated $n/2$ times, then the energy consumption of Eve is $\Theta(n^2)$, and the energy consumption of each of the remaining $n/2$ nodes is $\Omega(n \log n)$. In other words, if Eve can jam $d = \Theta(n^2)$ slots, then the energy consumption of a node might reach $\Omega(\sqrt{d} \log n)$. To overcome this difficulty, we let each node run one instance of standard binary backoff continuously, from the slot it joins the system to the slot it succeeds. This is a major change on the high-level structure of the algorithm, and it has important implications.

Particularly, on the one hand, the above modification enforces good energy complexity: assuming all nodes start simultaneously and the adversary jams the first d slots, the cost of each node in the first d slots would be only $\Theta(\log d)$. On the other hand, however, this simple tweak results in sub-optimal time complexity: if Eve jams the channel sufficiently long and then stops, it would take remaining nodes too many slots to send out their messages.

This is where the second channel comes into play. For each node, upon arrival, it will run a backoff procedure with more "aggressive" sending probabilities on channel two. As it turns out, $(c \log x)/x$-BACKOFF is the proper choice, where c is some large constant. Together with channel one, these two backoff procedures guarantee good time complexity regardless of the jamming pattern. In particular, Eve can jam sufficiently long to let the contention of both channels become $o(1)$, but through careful analysis we show in such a case channel two can still generate successes sufficiently often, at least in an amortized sense, thus enforcing the desired $O(n \log n + d)$ total time complexity. (It is worth noting, although $\Theta(\log x)/x$-BACKOFF is also utilized in [7, 13], the purposes are very different: in previous works, $\Theta(\log x)/x$-BACKOFF is used to generate "signals" so that nodes arriving at different times can group into "batches", whereas in this paper the procedure is used to ensure sufficiently frequent successful transmissions. $\Theta(\log x)/x$-BACKOFF has also been used for this latter purpose in [15].)

Interestingly, the above two-channel scheme works even if nodes are injected over time by the adversary (though the analysis becomes harder), so the only remaining issue is to let it work in the single-channel model. Notice that if nodes can access global slot indices, then a simple solution exists: all odd slots correspond to the first channel, and all even slots correspond to the second channel. Unfortunately, we do not assume nodes can access such information. Instead, we use a "synchronization procedure" inspired by [7]. For each node u, upon arrival, it will first run the synchronization procedure; when the synchronization procedure ends, all nodes that are in the system reach agreement on the parity of slots, so u can start running the two-channel algorithm (and other existing nodes resume running the two-channel algorithm).

Overview of algorithm for the static scenario. In this case, we gain the advantage that all nodes start simultaneously. This allows us to easily convert any two-channel algorithm into a corresponding single-channel algorithm, as all nodes agree on slot indices. Nonetheless, in this case, we are also targeting an improved $O(n+d)$ time complexity, thus the dynamic algorithm is inadequate.

Recall that when collision detection is not available and jamming is not present, to achieve constant throughput, previous works rely on repeating the "efficient part" of the binary exponential backoff procedure. Specifically, imagine there are n nodes running $1/x$-BACKOFF on one channel—called the data channel, and these nodes also run $(c \log x)/x$-BACKOFF on another channel—called the control channel. Then it can be shown, the first success on the control channel will happen in slot $\Theta(n)$. By then, remaining nodes will restart $1/x$-BACKOFF on the data channel, and restart $(c \log x)/x$-BACKOFF on the control channel.

Our first observation is that restarting the two backoff procedures from scratch is not necessary. Indeed, if n nodes run $1/x$-BACKOFF and $(c \log x)/x$-BACKOFF on two channels, then in the first n slots only limited number of successes would occur due to high contention. Hence, in our static algorithm, each node maintains a variable ℓ to control its sending probability: in each slot, each node sends on the data channel with probability $1/\ell$ and sends on the control channel with probability $(c \log \ell)/\ell$. Initially $\ell = 1$, then in each of the following slots, if the control channel does not generate a success, each node increases ℓ by one; otherwise each node halves the value of ℓ. Compared with the scheme used in previous works (which is resetting $\ell = 1$ after each control channel success), this new scheme only doubles the sending probability upon seeing a control channel success, efficiently maintaining the contention of the data channel within a desirable interval. (Notice that here $\Theta(\log x)/x$-BACKOFF is used to generate "signals" every some time to restart backoff procedures, and not intended for generating a large amount of successes. This is also the reason we call the second channel as "control channel" in the static scenario.)

Unfortunately, this doubling scheme still suffers poor energy efficiency if Eve focuses on jamming slots where the data channel contention is $\Theta(1)$. Nevertheless, a critical observation and an important advantage of this doubling scheme is, it only suffers poor energy efficiency if Eve jams at least $\Omega(n \log n)$ slots, which implies $d = \Omega(n \log n)$. But once $d = \Omega(n \log n)$, the time complexity we are targeting—which is $O(n+d)$—is dominated by d. In other words, when $d = \Omega(n \log n)$, after running the above doubling scheme for a while, the remaining nodes can switch to running another algorithm. This phase two algorithm can afford $\Theta(d)$ running time (which might be $\omega(n)$), in exchange for good energy efficiency.

So the condition for the transition from phase one to phase two is crucial. In the final algorithm, we come up with a simple criterion that works for arbitrary d values, without any prior knowledge on the value of d. Specifically, whenever a control channel success occurs, the value of ℓ (after halving) is recorded. Nodes use a variable m to maintain the minimum of these recorded values. By the end of each slot t, if $m \leq t/\log t$, remaining nodes will switch to phase two, which is to run a simple $(c \log x)/x$-BACKOFF from scratch on the control channel. (Here, $\Theta(\log x)/x$-BACKOFF is used to generate successful transmissions; i.e., similar to its role in the dynamic case algorithm.) In Section 4, we will explain the effectiveness of this transition condition in detail.

Lower bounds. For any robust communication algorithm that does not utilize collision detection, before the first successful message transmission, nodes cannot differentiate the following cases: (a) they have small sending probabilities, creating a contention too low; (b) they have large sending probabilities, creating a contention too high; or (c) the contention is right but the adversary is jamming. This dilemma suggests, if nodes want to achieve a good time complexity (even for creating the first success), they have to account for the first possibility and send sufficiently often upon arrival. Exploiting this observation allows us to prove a key technical lemma which connects the energy complexity and the time complexity of contention resolution algorithms.

Notice that the static scenario has trivial time complexity lower bound $\Omega(n + d)$. As for the dynamic scenario, the $\Omega(n \log n + d)$ time complexity bound is a direct corollary of Theorem 1.3 of [13]. Lastly, to obtain the energy complexity lower bounds, we combine the time complexity lower bounds and the aforementioned lemma.

Concentration inequalities. We conclude this section by introducing two concentration inequalities that will be frequently used. The first one is the so-called "convenient" Chernoff bound, which can be found in various textbooks on randomized algorithms (such as [23]). The second one is Lemma 3.4 from [13], which in turn relies on Lemma 3 of [7]. It will be used in an amortized argument that appears multiple times in the analysis.

LEMMA 2.3 (CHERNOFF BOUND). *Suppose $X_1, X_2, \cdots, X_N$ are N independent indicator random variables such that $\Pr[X_i = 1] = p_i$ for all $1 \leq i \leq N$. Let $X = \sum_{i=1}^{N} X_i$, then we have:*

- $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbb{E}[X]}{3}\right)$ *for any* $0 < \delta \leq 1$
- $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbb{E}[X]}{2}\right)$ *for any* $0 < \delta < 1$
- $\Pr[X \geq R] \leq 2^{-R}$ *for any* $R \geq 6 \cdot \mathbb{E}[X]$

LEMMA 2.4 (LEMMA 3.4 OF [13]). *Suppose $X_1, X_2, \cdots, X_N$ are N (not necessarily independent) random variables such that $\Pr[X_i = t \mid X_1 = x_1, X_2 = x_2, \cdots, X_{i-1} = x_{i-1}] \leq 1/t^{\Omega(1)}$ holds for any sufficiently large t, any $1 \leq i \leq N$, and any values $x_1, x_2, \cdots, x_{i-1}$ of $X_1, X_2, \cdots, X_{i-1}$, then $\sum_{i=1}^{N} X_i = O(N)$ w.h.p. in N.*

3 THE DYNAMIC SCENARIO

We now present our algorithm for the case where nodes are injected over time by the adversary. We will first give a more through introduction of the algorithm in the two-channel model, then present the pseudocode, and finally proceed to the analysis. Due to space constraints, detailed discussion on how to convert the algorithm to work in the single-channel setting is provided in the full paper.

Recall the high-level structure of the dynamic two-channel algorithm introduced in Section 2. To enforce good energy efficiency, each node runs one instance of $1/x$-BACKOFF continuously on the first channel upon arrival. However, a downside of this mechanism is sub-optimal time complexity: if Eve jams for a sufficiently long time period and then stops, remaining nodes would take too many additional slots to succeed. To fix this issue, for each node, upon its arrival, we let it run one instance of $(c \log x)/x$-BACKOFF continuously on the second channel, where c is a large constant.

To understand why $\Theta(\log x)/x$-BACKOFF on the second channel helps reduce time complexity, consider the simpler case where all n nodes start simultaneously, and let us examine how the contention evolves in the above two-channel algorithm. In the first n slots, the contention on both channels are $\Omega(1)$ and limited successes would occur. This part is inherently inefficient but short, so overall it has no large impact on performance. Next, consider slots from $n+1$ to $\lambda n \log n$ where λ is some large constant. If Eve does not jam a constant fraction of these slots on channel one, then the first channel alone would allow all nodes to succeed by the end of slot $\lambda n \log n$ (see [5]). In such a case, the overall time complexity is $O(n \log n)$. If, on the contrary, Eve does jam at least constant fraction of slots from $n+1$ to $\lambda n \log n$ on channel one, then $d = \Omega(n \log n)$. In such a case, assume slot s_d is the first slot after slot $\lambda n \log n$ that is not jammed by Eve, then we know $d = \Omega(s_d)$. Consider a node u that is still active in slot s_d, its sending probability on the second channel is $(c \log s_d)/s_d$. Notice that the contention on channel two in slot s_d is at most $n \cdot (c \log(\lambda n \log n))/(\lambda n \log n) = O(1)$, as $s_d \geq \lambda n \log n$. Therefore, in slot s_d and the $\Theta(s_d)$ slots following slot s_d, the probability that u succeeds on channel two is at least $\Theta((\log s_d)/s_d)$. In other words, in interval $[s_d, 2s_d]$, if Eve does not jam a constant fraction of these slots on channel two, then u will halt by the end of slot $2s_d = O(d)$, with high probability in $s_d = \Omega(n \log n)$. A union bound implies this claim holds for every node that is still active in slot s_d. To sum up, informally, if the two-channel algorithm is executed for a duration sufficiently large in $\Omega(n \log n)$, and if Eve does not jam some constant fraction of these slots on at least one channel, then all nodes must have succeeded by the end of these slots. In other words, the time complexity of the two-channel algorithm is $O(n \log n + d)$. This would also imply the energy cost of each node is $O(\log^2(n \log n + d))$, which could also be expressed as $O(\log^2 n + \log^2 d)$.

The above discussion intuitively illustrates the effectiveness of our algorithm, later in the complete analysis we will show it provides similar guarantees even if the n nodes are injected dynamically

by the adversary. The high-level argument, which is a generalization of the above discussion, being: (a) the total number of slots in which the contention on the first channel is $\Omega(1)$ cannot be too large, as each node runs a backoff instance continuously; (b) during the time period in which the contention of the first channel is between $\Theta(1)$ and $\Theta(1/\log n)$, if $\Theta(n \log n)$ such slots are not jammed, all n nodes would succeed; and (c) for slots where the contention of the first channel is $O(1/\log n)$, Eve must have jammed sufficiently many slots previously to let such slots occur, thus though successful transmissions will not occur too frequently, the overall time complexity can still be bounded by an amortized argument.

3.1 Algorithm Description

Assuming nodes can access two independent communication channels, the algorithm for the dynamic case is very simple: for each node, in the i-th slot since its activation, send with probability $1/i$ on channel one, and send with probability $(c \log i)/i$ on channel two. Here, $c > 0$ is some sufficiently large constant.

Algorithm for each node in the dynamic scenario:

From the arriving slot, run $(1/x)$-BACKOFF on channel one, and run $(c \log x)/x$-BACKOFF on channel two.

3.2 Algorithm Analysis

In this subsection, we formally analyze the performance of the dynamic algorithm in the two-channel setting. We assume jamming on one channel for one slot costs one unit of energy, and Eve has a total energy budget of d. On the other hand, to facilitate the process that converts a two-channel algorithm to the single-channel setting, we extend Eve's ability on injecting nodes:

Definition 3.1 (h-INTERFERENCE). Let $h : \mathbb{N}^+ \to \mathbb{R}^+$ be a function. In the two-channel model, we say Eve has the ability of h-INTERFERENCE if she can inject additional nodes called *interference nodes*. For each interference node, on the first channel, the adversary can specify whether the node starts running h-BACKOFF from the slot the node is injected or from the slot following the slot the node is injected; for each interference node, on the second channel, the node always starts running h-BACKOFF from the slot it is injected. An interference node will halt and leave the system upon hearing a success on any channel.

In analyzing the dynamic scenario two-channel algorithm, we assume Eve has the ability of $(c \log x)/x$-INTERFERENCE. In particular, she can inject up to n interference nodes beside the n normal nodes. In such setting, the definition for the throughput and the energy complexity of a two-channel algorithm, as well as the definition for active slot, need to be adjusted accordingly.

Definition 3.2 (Throughput and Energy Complexity with $(c \log x)/x$-INTERFERENCE). Let $\mathcal{A}$ be the two-channel algorithm each normal node runs after arriving. Let $f, g : \mathbb{N}^+ \to \mathbb{R}^+$ be two functions.

- A slot is *active* if either some interference node or some normal node is still active in that slot.
- Algorithm $\mathcal{A}$ achieves (f, g)-*time-cost* if there exists a constant C such that for any integer $n, d \geq 1$ and any adaptive adversary with jamming budget d that injects n normal nodes and up to n interference nodes, the number of active slots is at most $C \cdot (f(n) + g(d))$, w.h.p. in $n + d$.
- Algorithm $\mathcal{A}$ achieves (f, g)-*energy-cost* if there exists a constant C such that for any integer $n, d \geq 1$ and any adaptive adversary with jamming budget d that injects n normal nodes and up to n interference nodes, the maximum number of broadcasting attempts a normal node or an interference node made is at most $C \cdot (f(n) + g(d))$, w.h.p. in $n + d$.

The following lemma reveals the fact that by allowing the adversary to $(c \log x)/x$-INTERFERENCE, we can convert a two-channel algorithm to work in the single-channel setting, with guarantees on both time complexity and energy complexity. We defer its proof to the full paper where we discuss the conversion in detail.

LEMMA 3.3. *Suppose there is a two-channel algorithm achieving $(n \log n, d)$-time-cost and $(\log^2 n, \log^2 d)$-energy-cost when the adversary can $(c \log x)/x$-INTERFERENCE and inject n interference nodes, then there is a single-channel algorithm achieving $(n \log n, d)$-time-cost and $(\log^2 n, \log^2 d)$-energy-cost.*

Thus, we only need to focus on analyzing the runtime of the two-channel algorithm when the adversary can $(c \log x)/x$-INTERFERENCE. Specifically, we intend to prove the following theorem.

THEOREM 3.4. *The two-channel algorithm achieves $(n \log n, d)$-time-cost and $(\log^2 n, \log^2 d)$-energy-cost, even if the adversary can $(c \log x)/x$-INTERFERENCE and inject n interference nodes.*

To prove the above theorem, we divide active slots into two categories according to the contention on the second channel—ones that such contention reaches 0.5 and ones that such contention is below 0.5—and provide bounds for each of them. Throughout the analysis, when calculating the contention on a channel, we sum up the broadcasting probabilities of both interference nodes and normal nodes. Due to space constraints, complete proofs of the lemmas stated in this subsection are provided in the full paper.

Slots with large contention. In this part, we count the number of active slots in which the contention of the second channel is at least 0.5. To facilitate presentation, we define *congest slots*.

Definition 3.5. We say a slot is a *congest slot* if at least one of the following events happens in that slot: (a) channel one is jammed by Eve; (b) the contention created by normal nodes on channel one is at least 1; and (c) there exists a (normal or interference) node on channel one that sends with probability at least 0.5.

With the above definition, we can further divide the slots we care (i.e., active slots in which the contention of the second channel is at least 0.5) into the following three categories: (a) congest slots; (b) slots in which at least one interference node is active; and (c) remaining slots not in the above two categories.

Intuitively, due to the fact that each normal node runs $1/x$-BACKOFF on the first channel upon arrival and that the jamming budget of the adversary is limited, an $O(n \log n + d)$ bound can be obtained on the number of slots that belong to the first category. As for the number of slots belonging to the last category, observe that in each such slot, the first channel is not jammed and no interference nodes are present; moreover, the contention of the first channel is both lower bounded (since the contention of the

second channel is lower bounded and the contention of the two channels differ by a logarithmic factor) and upper bounded (due to the assumption that this slot is not a congest slot). Therefore, for category three slots, successes are likely to occur frequently, implying the number of such slots is limited.

Bounding the number of category two slots is more involved. To that end, we introduce the following definition and lemma.

Definition 3.6. During an execution of the dynamic two-channel algorithm, define *interference intervals* $I_1 = [L_1, R_1]$, $I_2 = [L_2, R_2]$, ..., $I_k = [L_k, R_k]$ inductively as following. Define $R_0 = 0$. For any $i \in [k]$, L_i is the first slot after R_{i-1} where the adversary injects interference nodes; and R_i is the first slot since L_i in which there is a successful message transmission on any channel.

Lemma 3.7. *For any $i \in [k]$, denote the number of interference nodes injected during interference interval I_i as n'_i, and denote the number of congest slots in interval I_i as d'_i. Then for any positive integer $t > C^3(n'_i \log n'_i + d'_i)$ where C is a sufficiently large constant, for any fixed $\ell_1, \ell_2, ..., \ell_{i-1}$, we have:*

$$\Pr[\ |I_i| = t \ \mid\ |I_1| = \ell_1, |I_2| = \ell_2, ..., |I_{i-1}| = \ell_{i-1}\ ] \le 1/t^{\Omega(1)}.$$

In a nutshell, Lemma 3.7 states that any interference interval cannot be too long, and its proof employs the following strategy: when $t > C^3(n'_i \log n'_i + d'_i) > C^3 d'_i$, a large fraction of the t slots are not congest slots; similarly, when $t > C^3(n'_i \log n'_i + d'_i) > C^3 n'_i \log n'_i$, for a large fraction of the t slots, the contention created by the interference nodes on channel one is limited. Together, they enforce a good upper bound on the total contention of channel one, for a large fraction of the t slots. On the other hand, since at least one interference node is active in an interference interval, there is also a lower bound on the total contention of channel one throughout the t slots: namely, $(c \log t)/t$. Therefore, given sufficient slots with contention neither too high nor too low, a success will likely occur within t slots on channel one.

With Lemma 3.7, we can use an amortized argument to bound the number of slots in which some interference node is active (i.e., number of category two slots). Recall the strategies we discussed previously for bounding category one slots and category three slots, we are now ready to state the main technical lemma of this part.

Lemma 3.8. *The number of active slots in which the contention on the second channel is at least* 0.5 *is $O(n \log n + d)$, w.h.p. in $n + d$.*

Slots with small contention. We now count the number of active slots where the contention of the second channel is less than 0.5.

Throughout this part, we treat the slots in which the contention on channel two reaches 0.5 as jammed slots. Formally, we say an active slot is a *busy slot* if in that slot at least one channel is jammed by the adversary or in that slot the contention on channel two reaches 0.5. Assume there are d' busy slots throughout the entire execution, due to Lemma 3.8, we know $d' = O(n \log n + d)$.

To facilitate presentation, we further introduce the notation of *complete intervals*.

Definition 3.9. During an execution, the *active interval* of a (normal or interference) node is the interval between the node's arriving and leaving (both inclusive). We define *complete intervals* $I_1 = [L_1, R_1], I_2 = [L_2, R_2], ..., I_k = [L_k, R_k]$ inductively as following. Define $R_0 = 0$. For any $i \in [k]$, let O_i be the first active slot after R_{i-1}. Then, I_i is defined to be the union of the active intervals that intersect with O_i.

See Figure 1 for an example. By the above definition, all active slots of an execution are contained within the union of all complete intervals, so $\sum_{i=1}^{k} |I_i|$ is an upper bound on the total number of active slots. However, bounding $\sum_{i=1}^{k} |I_i|$ differs from the proof of Lemma 3.7. In particular, interference intervals do not intersect with each other, so in the proof of Lemma 3.7, we can bound the length of each interference interval, conditioned on any previous execution history. By contrast, complete intervals may overlap.

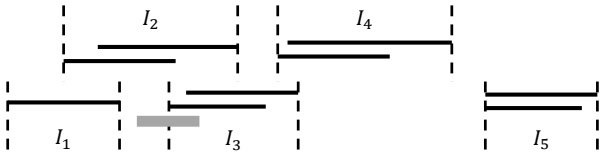

Figure 1: An example of complete intervals, where each solid horizontal line denotes the active interval of a node. Complete intervals with the same parity of index do not overlap with each other. Notice that not every active interval is necessarily included in some complete interval. For example, see the thick light-gray line in the figure.

Nonetheless, an important observation about complete intervals is that $I_1, I_3, I_5, \cdots$ are disjoint, so are $I_2, I_4, I_6, \cdots$. This can be verified easily: any node with active interval intersecting a slot after R_i must not intersect slot $R_{i-1} + 1$, otherwise it should be included in interval I_i, thus R_i should be larger, which is a contradiction. Hence, we can bound $|I_1| + |I_3| + |I_5| + \cdots$ and $|I_2| + |I_4| + \cdots$ separately.

Lemma 3.10. *For any $i \in [k]$, denote the number of injected (normal or interference) nodes during complete interval I_i as n_i, and denote the number of busy slots in I_i as d'_i. Then for any positive integer $t > C(n_i + d'_i)$ where C is a sufficiently large constant, for any odd i and any fixed $\ell_1, \ell_3, ..., \ell_{i-2}$, we have:*

$$\Pr[\ |I_i| = t \ \mid\ |I_1| = \ell_1, |I_3| = \ell_3, ..., |I_{i-2}| = \ell_{i-2}\ ] \le 1/t^{\Omega(1)}.$$

Similarly, for any even i and any fixed $\ell_2, \ell_4, ..., \ell_{i-2}$, we have:

$$\Pr[\ |I_i| = t \ \mid\ |I_2| = \ell_2, |I_4| = \ell_4, ..., |I_{i-2}| = \ell_{i-2}\ ] \le 1/t^{\Omega(1)}.$$

With the help of the above lemma, we can use an amortized argument to obtain the main technical lemma of this part.

Lemma 3.11. *Suppose during the execution of the two-channel algorithm there are a active slots in which the contention on the second channel is at least* 0.5. *Then, the total number of active slots is at most $O(a + n + d)$, with high probability in $n + d$.*

Proof sketch of Theorem 3.4. Combine Lemma 3.8 and Lemma 3.11, we know that for the two-channel algorithm, the total number of active slots is at most $O(n \log n + d)$. Since each (normal or interference) node runs some backoff procedure continuously on each channel, the above time complexity bound implies the total contention created by each node during its entire life cycle is at most $O(\log^2(n \log n + d))$. Recall that the total contention created by a node is also the expected number of times it broadcast during its life cycle, thus we can also derive the desired energy complexity bound. Again, see the full paper for complete proof.

We conclude this section by noting that combining Lemma 3.3 and Theorem 3.4 immediately gives Theorem 1.2, which states the time complexity and the energy complexity of our dynamic algorithm in the single-channel setting. (Complete proof of Lemma 3.3 is given in the full version of the paper.)

4 THE STATIC SCENARIO

In this section, we introduce our algorithm for the scenario where all nodes start execution simultaneously. Similar to Section 3, we first give a more through discussion on the intuition of the algorithm, then present the pseudocode, and finally proceed to the analysis.

In the static scenario, all nodes agree on slot indices, so the two-channel model is easy to implement: odd slots correspond to the first channel, which we also refer to as the "data channel"; and even slots correspond to the second channel, which we also refer to as the "control channel".

Assume indeed nodes can access two channels, recall the two-phase static algorithm we have discussed in Section 2. Specifically, in the first phase, nodes uses a variable ℓ to control its sending probability: in each slot, send with probability $1/\ell$ on the data channel, and send with probability $(c \log \ell)/\ell$ on the control channel. Initially $\ell = 1$. In each slot, if a control channel success occurs then $\ell = \ell/2$, effectively doubling the sending probability; otherwise, simply update $\ell = \ell + 1$. This phase one algorithm efficiently maintains the contention of the data channel within a desirable interval, enforcing a total time complexity of $O(n + d)$.

However, the above phase one algorithm could suffer poor energy efficiency when jamming from the adversary is strong. As a result, in the final algorithm, when nodes realize the adversary has jammed a lot, they will switch to a phase two algorithm, which is simply running $(c \log x)/x$-BACKOFF on the control channel. Assuming n' nodes remain when phase two starts, the energy complexity of each such node during phase two will be $O(\log^2 n' + \log^2 d)$, and the time complexity of phase two will be $O(n' \log n' + d)$. If we want to enforce the static algorithm's total time complexity to be $O(n + d)$, then we require $n' \log n' = O(n + d)$. Therefore, the transition from phase one to phase two is crucial: it cannot happen too late, otherwise phase one would incur large energy cost; it also cannot happen too early, otherwise we might have $n' \log n' = \omega(n + d)$, resulting in sub-optimal total time complexity.

Interestingly, a simple criterion exists. Upon arrival, each node initializes a variable called m and sets $m = \infty$. During phase one, whenever a control channel success occurs and ℓ is halved, nodes update $m = \min\{m, \ell\}$. By the end of each slot t, if $m \le t/\log t$, remaining nodes will switch to phase two. Next, we explain in detail why this criterion is the proper transition condition.

The value of m is an asymptotic upper bound on the number of remaining nodes. In phase one, the contention of the control channel will increase (particularly, double) only if a success occurs on the control channel. Moreover, due to Lemma 2.1, with high probability in ℓ a success will not happen when the contention of the control channel reaches $\Omega(\log \ell)$. Hence, the algorithm restricts the contention of the control channel to be $O(\log \ell)$. Recall that the contention of the two channels differ by a $\Theta(\log \ell)$ factor, thus the contention of the data channel is restricted to be $O(1)$ in phase one. On the other hand, by definition, the contention of the data channel is $n' \cdot (1/\ell)$ where n' is the number of remaining nodes. As a result, $n' = O(\ell)$ always holds throughout phase one. Recall that m is always the minimum value of ℓ up to now and the number of remaining nodes only decrease over time, we conclude m is always an asymptotic upper bound on the number of remaining nodes.

Energy complexity of phase one is guaranteed. Consider phase one of the algorithm. Assume it takes T_a slots for the first control channel success to occur. Then, for each slot after the first control channel success, $\Theta((\log m)/m)$ is an upper bound on nodes' sending probabilities as by then $m = O(\ell)$ always hold. Recall that for each slot t in phase one, $m = \Omega(t/\log t)$. Moreover, m only decreases overtime. As a result, if phase one ends by the end of slot T, then for each slot in $[T_a + 1, T]$, each node's sending probability is $O(\log(T/\log T)/(T/\log T)) = O((\log^2 T)/T)$. This implies the expected energy cost of each node for phase one is $O(\log^2 T_a + (T - T_a) \cdot (\log^2 T)/T) = O(\log^2 T)$. Recall that phase one lasts for at most $O(n + d)$ slots (otherwise all nodes would have succeeded), we conclude the energy cost of each node during phase one is $O(\log^2(n + d)) = O(\log^2 n + \log^2 d)$.

Time complexity of phase two is guaranteed. Due to the above discussion, we have $n' = O(m) = O(T/\log T)$, where n' is the number of nodes entering phase two and T is the duration of phase one. Moreover, $T = O(n + d)$ as phase one has time complexity $O(n + d)$. Therefore, we conclude $n' \log n' = O(n + d)$, implying the time complexity of phase two is $O(n' \log n' + d) = O(n + d)$.

In summary, $m = \Omega(t/\log t)$ guarantees the energy complexity of phase one, while $m = O(t/\log t)$ guarantees the time complexity of phase two. Since m is decreasing over time and t is increasing over time, the first time $m \le t/\log t$ is satisfied is the proper moment to transition from phase one to phase two.

4.1 Algorithm Description

Algorithm for each node in the static scenario:

Phase I: Initialize $\ell \leftarrow 1$, $t \leftarrow 1$, $m \leftarrow \infty$. Repeat the following in each slot until $m \le t/\log t$, or the node succeeds (and halts).

(1) Send with probability $\min\{1/\ell, 1\}$ on the data channel, and send with probability $\min\{(c \log \ell)/\ell, 1\}$ on the control channel, where c is a sufficiently large constant.
(2) If in this slot the control channel generates a success, then let $\ell \leftarrow \max\{\ell/2, 1\}$ and let $m \leftarrow \min\{m, \ell\}$; otherwise let $\ell \leftarrow \ell + 1$.
(3) $t \leftarrow t + 1$.

Phase II: Run a fresh instance of $(c \log x)/x$-BACKOFF on the control channel, until the node succeeds (and halts).

Our static algorithm is shown above. It is described in the two-channel model for simplicity, and can be easily converted to the single-channel setting.

4.2 Algorithm Analysis

Due to space constraints, complete proofs for the lemmas stated in this subsection are in the full paper. Nevertheless, we will often provide a sketch to allow readers to grasp the high-level idea.

We first argue the energy consumption of each node is poly-

logarithmic in its total running time. This is because the life cycle of each node can be divided into (at most) three parts: the first part ends when first control channel success occurs, the second part ends when phase one ends, and the third part ends when the node succeeds. The energy consumption during part one and part three are obvious, since in each of the two parts the node runs some backoff procedures continuously. Obtaining the energy consumption for part two relies on the property that our algorithm ensures an upper bound on each node's broadcasting probability (as a function of the value of m) during phase one.

Lemma 4.1. *Each node running the static algorithm sends $O(\log^2 t)$ times in the first t slots, with high probability in t.*

In the reminder of this subsection, we inspect the time complexity of our static algorithm, and we do so by proving a bound for phase one and phase two separately.

We will first focus on phase two since it is easier to analyze. In particular, if a node u runs phase two for $t = \Omega(n' \log n' + d)$ slots where n' is the number of nodes entering phase two, then most of these slots are not jammed (since $d = O(t)$). Moreover, among the non-jammed slots, there will also exist many slots in which the contention created by nodes beside u is bounded, and in each such slot u broadcasts with probability at least $(c \log t)/t$. Therefore, when $t = \Omega(n' \log n' + d)$, node u will likely to succeed within t slots during phase two. A union bound then implies this holds for all nodes entering phase two. In summary, we claim:

Lemma 4.2. *Denote the duration of phase two as T, denote the number of nodes that join phase two as n'. Then for any $t \geq c^2(n' \log n' + d)$, we have $\Pr[T = t] \leq 1/t^{\Omega(1)}$.*

Next, we analyze the time complexity of phase one. To that end, we define *complete intervals*. Specifically, a complete interval during phase one is the time interval between two (consecutive) control channel successes. Clearly, phase one can be partitioned into a set of complete intervals.

The following lemma shows the first complete interval will not be too long. Its proof strategy is somewhat similar to that of Lemma 4.2. Specifically, a large fraction of the first $\Theta(n + d)$ slots of phase one are not jammed. Among many of these non-jammed slots, the contention on the control channel has both a proper lower bound and a proper upper bound. As a result, a success will likely occur within these $\Theta(n + d)$ slots.

Lemma 4.3. *In phase one, the first control channel success happens in $O(n + d)$ slots, with high probability in $n + d$.*

Then, we bound the duration of the complete intervals after the first control channel success, since these complete intervals can be amortized by the number of *congest slots* defined as below.

Definition 4.4. In phase one, we say a slot is a *congest slot* if it is jammed by the adversary (on either channel), or the contention of the data channel in this slot is at least $1/c^2$.

The following lemma shows the total number of congest slots cannot be too large. To prove it, we first argue, during phase one, for each slot after the first control channel success and before slot $(n + d)^2$, the contention on the control channel is $O(\log(n + d))$. Then, we argue that during this time interval, the value of ℓ satisfies $\log \ell = \Omega(\log(n + d))$. Since the contention of data channel and the contention of control channel differs by a $\Theta(\log \ell)$ factor, we conclude the contention on data channel during this interval is $O(1)$. Now, recall that a congest slot means the contention on data channel is at least $1/c^2$. This means during phase one, after the first control channel success, in each non-jammed congest slot, a success will occur with constant probability. Hence, all nodes will succeed within $O(n)$ non-jammed congest slots after the first control channel success. Since Eve jams at most d slots, we conclude there are $O(n + d)$ congest slots, which is the lemma statement.

Lemma 4.5. *In phase one, the number of congest slots in the first $(n + d)^2$ slots is $O(n + d)$, with high probability in $n + d$.*

At this point, we are ready to formally define complete interval and use an amortized argument to bound its length. Notice that in proving the following lemma, besides congest slots, we also use the change in the value of ℓ to amortize interval length.

Lemma 4.6. *Suppose k control channel successes happen during phase one, and the i-th success happens in slot t_{i+1}. Define $t_1 = 0$, and define t_{k+1} be the last slot of phase one. For every $i \in [k]$, define complete interval $I_i = [t_i + 1, t_{i+1}]$, define d'_i to be the number of congest slots during I_i, and define $\Delta\ell_i$ to be the value of ℓ at the beginning of slot t_i+1 minus the value of ℓ at the end of slot t_{i+1}. Then, for any positive integer $t > C(d'_i+\Delta\ell_i)$, where C is a sufficiently large constant, we have $\Pr[\ |I_i| = t \ \mid \ |I_1| = T_1, |I_2| = T_2, ..., |I_{i-1}| = T_{i-1}\ ] \leq 1/t^{\Omega(1)}$, for any fixed $T_1, T_2, ..., T_{i-1}$.*

We are now ready to prove the guarantees enforced by our static algorithm.

Proof of Theorem 1.3. We first bound the length of phase one. Assume k control channels successes occur during phase one, so phase one can be divided into k complete intervals. For every $i \in [k]$, let T_i be a random variable such that $T_i = |I_i|$ if $|I_i| > C(d'_i+\Delta\ell_i)$, otherwise $T_i = 0$. (Recall $I_i, C, d'_i, \Delta\ell_i$ are defined in Lemma 4.6.) For every $i \in [k+1, n+d]$, define $T_i = 0$. Thus, the duration of phase one is $\sum_{i=1}^{k} |I_i| \leq \sum_{i=1}^{k}(T_i + C(d'_i + \Delta\ell_i)) = \sum_{i=1}^{k} T_i + C \cdot \sum_{i=1}^{k}(d'_i + \Delta\ell_i) = \sum_{i=1}^{n+d} T_i + C \cdot \sum_{i=1}^{k}(d'_i + \Delta\ell_i)$. Due to Lemma 4.6 and Lemma 2.4, $\sum_{i=1}^{n+d} T_i = O(n + d)$, w.h.p. in $n + d$. Due to the analysis in the proof of Lemma 4.5, $\sum_{i=1}^{k} d'_i = O(n + d)$, w.h.p. in $n + d$. Lastly, notice $\Delta\ell_i = \ell_{t_i+1} - \ell_{t_{i+1}}/2$ and $\ell_{t_{i+1}+1} = \ell_{t_{i+1}}/2$, hence $\sum_{i=1}^{k} \Delta\ell_i = \ell_{t_1+1} - \ell_{t_{k+1}+1}/2 \leq \ell_{t_1+1} = 1$. Therefore, length of phase one is $O(n + d) + C \cdot O(n + d) + C \cdot 1 = O(n + d)$, w.h.p. in $n + d$.

Next, we bound the length of phase two, and we consider two complement cases depending on the relationship between n and d. The first case is $d = \Omega(n \log n)$. According to Lemma 4.2, in such a case, the second phase runs for $O(n \log n + d)$ slots, w.h.p. in $n + d$. The other case is $d = O(n \log n)$, and its analysis is more involved. (Notice, when $d = O(n \log n)$, if an event happens w.h.p. in n, then it also happens w.h.p. in $n + d$.)

Assume $d = O(n \log n)$, we first prove the claim that in phase one the first control channel success happens after time slot n, with high probability in n. To see this, notice that in phase one, for each slot before slot n and before the first control channel success, the contention on the control channel is at least $n \cdot (c \log n)/n \geq c \log n$. Hence, due to Lemma 2.1, with high probability in n, a control channel success will not occur in this slot. Take a union bound over

the first n slots in phase one, the claim is proved.

Then we argue, in phase one, for any slot after the first control channel success and before slot $(n+d)^2$, the contention of the data channel in that slot is $O(1)$, with high probability in n. To see this, notice that due to Claim 4.5.1 (see page 17 of the full paper) and the assumption $d = O(n \log n)$, in phase one, for any slot after the first control channel success and before slot $(n+d)^2$, the contention of the control channel in that slot is $O(\log n)$. On the other hand, in phase one, if the first control channel success happens after slot n (this happens with high probability in n due to the discussion in the last paragraph), then by algorithm description, $\ell \geq n/\log n$ always holds after the first control channel success. (Otherwise, phase one will end.) Lastly, notice that in phase one, for each slot after the first control channel success, the contention on the data channel and the control channel differs by a factor of $c \log \ell$, which is at least $\Omega(\log n)$ since we have shown $\ell \geq n/\log n$. At this point, we can conclude, in phase one, for any slot after the first control channel success and before slot $(n+d)^2$, the contention of the data channel in that slot is $O(1)$, with high probability in n.

Now, suppose phase one ends by the end of slot T and n' nodes remain. Suppose the value of ℓ at the beginning of T is ℓ_T. By algorithm description, we know $\ell_T/2 \leq T/\log T$. Since in slot T each node sends on the data channel with probability $1/\ell_T$ and we have shown above with high probability in n the contention on the data channel is $O(1)$ in that slot, we know $n' = O(1)/(1/\ell_T) = O(\ell_T) = O(T/\log T)$ with high probability in n. Recall we have already shown the time complexity of phase one—which is T—is $O(n+d)$, with high probability in $n+d$. Hence, when $d = O(n \log n)$, by Lemma 4.2, the time complexity of phase two is $O(n' \log n' + d) = O(n+d)$, with high probability in n.

At this point, we have proved the total time complexity of the static algorithm is $O(n+d)$, with high probability in $n+d$. Finally, recall Lemma 4.1, we can also conclude the energy complexity of each node is $O(\log^2 n + \log^2 d)$, with high probability in $n+d$. □

5 LOWER BOUNDS

In this part, we prove several complexity lower bounds for the contention resolution problem, when collision detection is not available and external interference is present. These bounds show both our algorithms achieve optimal time complexity. As for energy complexity, the dynamic algorithm also exactly matches the lower bound, whereas the static algorithm misses the lower bound by a poly-logarithmic factor regarding the dependency on n (and matches the lower bound regarding the dependency on d). In particular, in the static scenario, the lower bound is $\Omega(\log\log n + \log^2 d)$, while our algorithm incurs a per-node energy cost of $O(\log^2 n + \log^2 d)$.

Before proving the lower bounds, we first introduce the following key technical lemma. Recall the discussion in Section 2, this lemma helps us to connect the time complexity and the energy complexity of a contention resolution algorithm.

Lemma 5.1. *For any function $f : \mathbb{N}^+ \to \mathbb{R}^+$, if algorithm $\mathcal{A}$ achieves $(f(n), g(d))$-time-cost in the static case with $g(d) = d$, and if algorithm $\mathcal{A}$ does not observe any success in the first t slots, then in expectation algorithm $\mathcal{A}$ sends $\Omega(\log^2 t)$ times in the first t slots.*

Proof. Consider the case one node u runs $\mathcal{A}$. Since $\mathcal{A}$ achieves $(f(n), d)$-time cost, there exists a constant C such that the number of active slots—i.e., the time required for u to succeed—is at most $C(f(1)+d)$, w.h.p. in d. Here, d is the number of slots jammed by Eve. Now, consider the first t slots, assume Eve uses the following strategy. She jams first $t/(4C)$ slots; she also jams another $t/(4C)$ slots chosen uniformly at random from interval $(t/(4C), t]$. In this scenario, by the end of slot t, we have $C(f(1)+d) = Cf(1) + t/2 < t$ for any $t > 2Cf(1)$. Hence, there is at least one success in the first t slots, w.h.p. in d. Since $d = \Theta(t)$, this claim also holds w.h.p. in t.

Denote $k(t)$ as the number of times node u broadcasts in interval $(t/(4C), t]$. Since in interval $(t/(4C), t]$ Eve randomly chooses $t/(4C)$ slots to jam, the probability that no success occurs in these slots is at least $\sum_{k \leq \frac{t}{8C}} \frac{\Pr[k(t)=k]}{(8C)^k}$. Recall there must exist a success in these slots with probability at least $1 - 1/t$, so $\sum_{k \leq \frac{t}{8C}} \frac{\Pr[k(t)=k]}{(8C)^k} \leq \frac{1}{t}$. Since $\frac{t}{8C} \geq \log_{8C} \frac{t}{2}$ for sufficiently large t, we have:

$$\sum_{k \leq \frac{t}{8C}} \frac{\Pr[k(t)=k]}{(8C)^k} \geq \sum_{k \leq \log_{8C} \frac{t}{2}} \frac{\Pr[k(t)=k]}{(8C)^k} \geq \sum_{k \leq \log_{8C} \frac{t}{2}} \frac{\Pr[k(t)=k]}{(8C)^{\log_{8C} t/2}}$$
$$\geq \sum_{k \leq \log_{8C} \frac{t}{2}} \frac{\Pr[k(t)=k]}{t/2} = \frac{\Pr\left[k(t) \leq \log_{8C}(t/2)\right]}{t/2}$$

If $\Pr\left[k(t) \leq \log_{8C} \frac{t}{2}\right] > 1/2$, then $\sum_{k \leq \frac{t}{8C}} \frac{\Pr[k(t)=k]}{(8C)^k} > \frac{1/2}{t/2} = \frac{1}{t}$, which is a contradiction. Therefore, $\Pr\left[k(t) > \log_{8C} \frac{t}{2}\right] \geq 1 - 1/2 = 1/2$, implying $\mathbb{E}[k(t)] \geq \frac{1}{2} \log_{8C} \frac{t}{2}$.

By a similar argument, we know the expected number of times u broadcasts in interval $\left(\frac{t}{(4C)^{i+1}}, \frac{t}{(4C)^i}\right]$ is at least $\frac{1}{2} \log_{8C} \frac{t/2}{(4C)^i}$, for any $1 \leq i \leq l$ where $l = \log_{4C} t$. Therefore, in total the expected number of times u broadcasts in the first t slots is at least:

$$\sum_{0 \leq i \leq l} \frac{1}{2} \log_{8C} \frac{t/2}{(4C)^i} \geq \sum_{0 \leq i \leq l/2} \frac{1}{2} \log_{8C} \frac{t/2}{(4C)^{l/2}} = \Omega\left(\log^2 t\right)$$

□

Since an algorithm with $(f(n), g(d))$-time cost in the dynamic case is also an algorithm with $(f(n), g(d))$-time cost in the static case, the above lemma also holds for dynamic algorithms.

We are now ready to prove the lower bounds, and we start by considering the dynamic case.

Proof of Theorem 1.4. Notice that $f_t(n) = \Omega(n \log n)$ is implied by Theorem 1.3 of [13]. In the reminder of the proof, we focus on the energy complexity.

We first prove $g_e(d) = \Omega(\log^2 d)$. Consider a simple adversary strategy that injects one node in the first slot and jams the first d slots, then there are no successes in the first d slots. According to Lemma 5.1, the injected node will send in expectation $\Omega(\log^2 d)$ times in those slots. Suppose $g_e(d) = o(\log^2 d)$, then with high probability in d, the node sends $o(\log^2 d)$ times in the first d slots, this leads to an expectation of at most $o(\log^2 d) + d \cdot 1/d^{\Omega(1)} = o(\log^2 d)$ times of sending in the first d slots, which is a contradiction.

Then we prove $f_e(n) = \Omega(\log^2 n)$. Consider the adversary strategy that injects $\sqrt{n}$ nodes in each of the first $\sqrt{n}$ slots. Without loss of generality, assume the algorithm instructs each node to send in the first slot (after arriving) with probability $x_1 > 0$. Then, for each of the first $\sqrt{n}$ slots, the contention is at least $x_1\sqrt{n}$. Since the algorithm does not know the value of n, the value of x_1 cannot

depend on the value of n; this implies $x_1\sqrt{n} \geq n^{0.4}$ when n is sufficiently large. Therefore, due to Lemma 2.1 and the union bound, for sufficiently large n, in the first $\sqrt{n}$ slots, with high probability in n no success will occur. Consider a node u that is injected in the first slot. Due to Lemma 5.1, we can further conclude, in the first $\sqrt{n}$ slots, in expectation u sends at least $\Omega(\log^2\sqrt{n}) = \Omega(\log^2 n)$ times. Now, if $f_e(n) = o(\log^2 n)$, then with high probability in n, node u sends $o(\log^2 n)$ times in the first $\sqrt{n}$ slots; this leads to an expectation of at most $o(\log^2 n) + \sqrt{n} \cdot 1/n^{\Omega(1)} = o(\log^2 n)$ times of sending in the first $\sqrt{n}$ slots, which is a contradiction. □

Next, we consider the static case (i.e., Theorem 1.5). In such scenario, the $\Omega(n + d)$ time complexity bound is obvious, since in each slot there is at most one success (thus giving the $\Omega(n)$ bound) and the adversary can jam d slots to block any success (thus giving the $\Omega(d)$ bound). Therefore, the non-trivial part in Theorem 1.5 is the $\Omega(\log\log n + \log^2 d)$ energy complexity. Notice the proof for $g_e(d) = \Omega(\log^2 d)$ in the proof of Theorem 1.4 can be carried over to the static case without any modification, so what remains is to show $f_e(n) = \Omega(\log\log n)$. Due to space constraints, complete proof for $f_e(n) = \Omega(\log\log n)$ is deferred to the full paper. Nonetheless, the high-level strategy for proving it is similar to that of proving $f_e(n) = \Omega(\log^2 n)$ in the proof of Theorem 1.4: we first come up with a jamming strategy that ensures no node would succeed for a sufficiently long time period in the static case (regardless of the algorithm the nodes use), and then apply Lemma 5.1.

Finally, we note that our lower bounds are strong in the sense that they hold even for an oblivious adversary (i.e., an offline adversary). By contrast, our algorithms can tolerate an adaptive adversary. Moreover, since the lower bounds only consider the time and the energy required to generate the first success, it is likely that they also hold for the leader election problem in similar models.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62172207, and by the Jiangsu Provincial Department of Science and Technology (China) under Grant No. BK20211148.

REFERENCES

[1] Kunal Agrawal, Michael Bender, Jeremy Fineman, Seth Gilbert, and Maxwell Young. 2020. Contention Resolution with Message Deadlines. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '20)*. ACM, 23–35.

[2] David Aldous. 1987. Ultimate instability of exponential back-off protocol for acknowledgment-based transmission control of random access communication channels. *IEEE Transactions on Information Theory* 33, 2 (1987), 219–223.

[3] Lakshmi Anantharamu, Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2019. Packet latency of deterministic broadcasting in adversarial multiple access channels. *J. Comput. System Sci.* 99 (2019), 27–52.

[4] Baruch Awerbuch, Andrea Richa, and Christian Scheideler. 2008. A Jamming-Resistant MAC Protocol for Single-Hop Wireless Networks. In *Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC '08)*. ACM, 45–54.

[5] Michael Bender, Martin Farach-Colton, Simai He, Bradley Kuszmaul, and Charles Leiserson. 2005. Adversarial Contention Resolution for Simple Channels. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '05)*. ACM, 325–332.

[6] Michael Bender, Jeremy Fineman, Seth Gilbert, and Maxwell Young. 2018. Scaling Exponential Backoff: Constant Throughput, Polylogarithmic Channel-Access Attempts, and Robustness. *J. ACM* 66, 1 (2018), 33 pages.

[7] Michael Bender, Tsvi Kopelowitz, William Kuszmaul, and Seth Pettie. 2020. Contention Resolution without Collision Detection. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20)*. ACM, 105–118.

[8] Michael Bender, Tsvi Kopelowitz, Seth Pettie, and Maxwell Young. 2016. Contention Resolution with Log-Logstar Channel Accesses. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC '16)*. ACM, 499–508.

[9] Petra Berenbrink, Colin Cooper, and Zengjian Hu. 2009. Energy efficient randomised communication in unknown AdHoc networks. *Theoretical Computer Science* 410, 27 (2009), 2549–2561.

[10] Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, Qizheng He, Wenzheng Li, and Seth Pettie. 2018. The Energy Complexity of Broadcast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC '18)*. ACM, 95–104.

[11] Yi-Jun Chang, Wenyu Jin, and Seth Pettie. 2019. Simple Contention Resolution via Multiplicative Weight Updates. In *2nd Symposium on Simplicity in Algorithms (SOSA '19)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 16:1–16:16.

[12] Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. 2019. Exponential Separations in the Energy Complexity of Leader Election. *ACM Transactions on Algorithms* 15, 4 (2019), 31 pages.

[13] Haimin Chen, Yonggang Jiang, and Chaodong Zheng. 2021. Tight Trade-off in Contention Resolution without Collision Detection. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21)*. ACM, 139–149.

[14] Bogdan Chlebus, Leszek Gasieniec, Dariusz R. Kowalski, and Tomasz Radzik. 2005. On the Wake-Up Problem in Radio Networks. In *Proceedings of the 2005 International Colloquium on Automata, Languages, and Programming (ICALP '05)*. Springer Berlin Heidelberg, 347–359.

[15] Gianluca De Marco and Grzegorz Stachowiak. 2017. Asynchronous Shared Channel. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC '17)*. ACM, 391–400.

[16] Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, Fabian Kuhn, and Calvin Newport. 2009. The Wireless Synchronization Problem. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing (PODC '09)*. ACM, 190–199.

[17] Seth Gilbert and Dariusz R. Kowalski. 2010. Trusted Computing for Fault-Prone Wireless Networks. In *Proceedings of 2010 International Symposium on Distributed Computing (DISC '10)*. Springer Berlin Heidelberg, 359–373.

[18] Leslie Ann Goldberg, Philip D. Mackenzie, Mike Paterson, and Aravind Srinivasan. 2000. Contention Resolution with Constant Expected Delay. *J. ACM* 47, 6 (2000), 1048–1096.

[19] Jonathan Goodman, Albert G. Greenberg, Neal Madras, and Peter March. 1988. Stability of Binary Exponential Backoff. *J. ACM* 35, 3 (1988), 579–602.

[20] Johan Hastad, Tom Leighton, and Brian Rogoff. 1987. Analysis of Backoff Protocols for Multiple Access Channels. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC '87)*. ACM, 241–253.

[21] James Kurose and Keith Ross. 2017. *Computer Networking: A Top-Down Approach, 7th Edition.* Pearson.

[22] Gianluca De Marco, Dariusz Kowalski, and Grzegorz Stachowiak. 2019. Deterministic Contention Resolution on a Shared Channel. In *Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS '19)*. IEEE, 472–482.

[23] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis.* Cambridge University Press.

[24] Miguel Mosteiro, Antonio Fernandez Anta, and Jorge Ramon Muñoz. 2011. Unbounded Contention Resolution in Multiple-Access Channels. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC '11)*. ACM, 211–212.

[25] Joseph Polastre, Robert Szewczyk, and David Culler. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*. IEEE, 364–369.

[26] Raghu Ramakrishnan and Johannes Gehrke. 2002. *Database Management Systems, 3rd Edition.* McGraw-Hill.

[27] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2010. A Jamming-Resistant MAC Protocol for Multi-Hop Wireless Networks. In *Proceedings of the 2010 International Symposium on Distributed Computing (DISC '10)*. Springer Berlin Heidelberg, 179–193.

[28] Andrew Tanenbaum and Herbert Bos. 2014. *Modern Operating Systems, 4th Edition.* Pearson.

[29] Dan Willard. 1986. Log-Logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM J. Comput.* 15, 2 (1986), 468–477.