Fully Energy-Efficient Randomized Backoff: Slow Feedback Loops Yield Fast Contention Resolution

Michael A. Bender
Stony Brook University
Stony Brook, NY 11794-2424, USA
bender@cs.stonybrook.edu

Jeremy T. Fineman
Georgetown University
Washington, DC 20057-1232, USA
jfineman@cs.georgetown.edu

Seth Gilbert
National University of Singapore
Singapore 117417
seth.gilbert@comp.nus.edu.sg

John Kuszmaul
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Google
Cambridge, MA 02142, USA
john.kuszmaul@gmail.com

Maxwell Young
Mississippi State University
Mississippi State, MS 39762, USA
myoung@cse.msstate.edu

ABSTRACT

Contention resolution addresses the problem of coordinating access to a shared communication channel. Time is discretized into synchronized slots, and a packet transmission can be made in any slot. A packet is successfully sent if no other packet is also transmitted during that slot. If two or more packets are sent in the same slot, then these packets collide and fail. Listening on the channel during a slot provides ternary feedback, indicating whether that slot had (0) silence, (1) a successful transmission, or (2+) noise. No other feedback or exchange of information is available to packets. Packets are (adversarially) injected into the system over time. A packet departs the system once it is successfully sent. The goal is to send all packets while optimizing throughput, which is roughly the fraction of successful slots.

Most prior contention resolution algorithms with constant throughput require a short feedback loop, in the sense that a packet's sending probability in slot $t+1$ is fully determined by its internal state at slot t and the channel feedback at slot t. This paper answers the question of whether these short feedback loops are necessary; that is, how often must listening and updating occur in order to achieve constant throughput? We can restate this question in terms of energy efficiency: given that both listening and sending consume significant energy, is it possible to have a contention-resolution algorithm with ternary feedback that is efficient for both operations?

A shared channel can also suffer random or adversarial noise, which causes any listener to hear noise, even when no packets are actually sent. Such noise arises due to hardware/software failures or malicious interference (all modeled as "jamming"), which can have a ruinous effect on the throughput and energy efficiency. How does noise affect our goal of long feedback loops/energy efficiency?

Tying these questions together, we ask: *what does a contention-resolution algorithm have to sacrifice to reduce channel accesses?* Must we give up on constant throughput? What about robustness to noise? Here, we show that we need not concede anything by presenting an algorithm with the following guarantees. Suppose there are N packets arriving over time and $\mathcal{J}$ jammed slots, where the input is determined by an adaptive adversary. With high probability in $N+\mathcal{J}$, our algorithm guarantees $\Theta(1)$ throughput and polylog$(N+\mathcal{J})$ channel accesses (sends or listens) per packet. We also have analogous guarantees when the input stream is infinite.

CCS CONCEPTS

• **Theory of computation** → **Distributed algorithms**; • **Networks** → **Network algorithms**.

KEYWORDS

Contention resolution, backoff,energy efficiency, jamming

ACM Reference Format:
Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, John Kuszmaul, and Maxwell Young. 2024. Fully Energy-Efficient Randomized Backoff: Slow Feedback Loops Yield Fast Contention Resolution. In *ACM Symposium on Principles of Distributed Computing (PODC '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3662158.3662807

1 INTRODUCTION

Since the 1970s, randomized ***backoff*** protocols such as binary exponential backoff [48], have been used for managing contention on a shared communication channel. Originally used in the ALOHA system [1] and Ethernet [48], randomized backoff plays an important role in a wide range of applications, including WiFi [35], wireless sensor networks [39], transactional memory [34, 59], and congestion control [62]. The salient feature of the communication channel is that it supports only one message transmission at a time: if more than one message is sent simultaneously, there is a ***collision*** resulting in indecipherable noise [30, 31, 41, 48, 65].

This ***contention-resolution problem*** is formalized as follows. There are N packets arriving over time, and each packet needs to

PODC '24, June 17–21, 2024, Nantes, France

ACM ISBN 979-8-4007-0668-4/24/06
https://doi.org/10.1145/3662158.3662807

be successfully transmitted on the channel.[1] Time is divided into synchronized ***slots***, each of which is sized to fit a single packet. To be successfully sent, the packet requires exclusive access to the channel that is, the packet must be the only one transmitted during that slot. Otherwise, if two or more packets are transmitted in the same slot, the result is a collision, where none of the transmitted packets succeed. A packet departs the system once it is successfully sent. There is no *a priori* coordinator or central authority; packet transmissions are scheduled in a distributed manner. The objective is to send all packets while optimizing the ***throughput*** of the channel, which is roughly the fraction of successful slots.

A popular contention-resolution protocol is binary exponential backoff [48] (see [3–5, 32, 32, 37, 38, 58, 60]). Informally, under binary exponential backoff, a packet that has been in the system for t slots is sent with probability $\Theta(1/t)$.

Feedback loops: short versus long (versus no feedback). An elegant, but ultimately problematic, feature of classical binary exponential backoff is that it is ***oblivious***—packets remaining in the system do not use channel feedback to adjust their behavior: a packet with age t sends with probability $\Theta(1/t)$ until it succeeds, regardless of channel history. The unfortunate result is that with adversarial packet arrivals, binary exponential backoff supports only a subconstant throughput—specifically, $O(1/\log N)$; in fact, even for the batch case where all N packets arrive at the same time, the throughput of binary exponential backoff is only $O(1/\log N)$ [8].

In contrast, contention-resolution protocols can exploit frequent channel feedback to achieve $\Theta(1)$ throughput under adversarial arrivals [7, 10–12, 14]. These protocols are not oblivious—packets ***listen*** on the channel and adjust their sending probabilities up or down based on this feedback.

The primary model for channel feedback is the ***ternary-feedback model*** [2, 6, 10, 19–21, 28, 29, 33, 40]. In this model, a packet can listen on the channel in each slot and learn whether that slot is (0) ***empty***, if no packets send, (1) ***successful***, if exactly one packet sends, or (2+) ***noisy***, if two or more packets send. Based on this ternary feedback, the packet can decide when to attempt to send next. Once a packet succeeds, it immediately departs the system.

Most constant-throughput algorithms with ternary-feedback (e.g., [7, 10–12, 14, 25, 52, 55–57]) listen on the channel in every slot (or every constant number of slots). That is, these algorithms have a short feedback loop: a packet's sending probability in slot $t+1$ is fully determined by its internal state at slot t and the channel feedback at slot t. For example, the algorithm by Chang, Jin, and Pettie [14], listens in every slot t and multiplicatively updates the sending probability in slot $t+1$ based on whether it heard silence, a successful transmission, or a collision in slot t.

A key question is as follows: *how frequently does the packet need to listen on the channel and update its behavior in order to achieve constant throughput?* In every slot? In a vanishingly small fraction of slots? How short a feedback loop is necessary for good throughput?

[1] For ease of exposition, we slightly abuse terminology and have the packets themselves taking action (e.g., sending themselves on the channel, listening on the channel), rather than introducing "agents"/"devices"/"senders", where each one arrives on the scene with a packet to transmit.

As an analogy, one cannot navigate a ship without a control feedback loop: monitoring the surroundings and correcting course. One option is to constantly monitor and continuously update the heading to avoid obstacles. But the analogous question is whether one can still safely navigate with only a vanishingly small amount of course correction. In contention resolution, the monitoring is via the channel sensing and the course being corrected is the transmission probabilities.

Robustness and noise. Finally, in addition to the factors discussed above, much of the recent work on contention resolution [2, 6, 7, 10, 14, 16, 18, 36, 52, 55–57] has sought to address an additional factor, which is that the real world is often noisy. Sometimes interference prevents transmissions in a slot and listeners hear noise even if nobody actually sends. Noisy channels arise due to hardware/software failures, co-located devices, or malicious jamming [17, 45, 49, 53, 66]. Regardless of the source of the noise in a slot, we may think of these slots as being "jammed" by an adversary. Jamming has evolved from a mostly-theoretical risk into a credible threat to systems over the past decade, with several publicized examples [23, 50, 51, 61].

After a long line of work, there are now many contention-resolution protocols that achieve constant throughput in the presence of noise (e.g. [7, 10, 14, 52, 55–57]); however, again, these protocols listen to the channel in every slot and update their sending probabilities accordingly. Our goal is to eliminate short feedback loops not just in the classical model, but also with jamming.

Minimizing channel accesses = energy efficiency. Up until now, we have discussed whether one can minimize listening and thus avoid providing immediate feedback for a contention-resolution algorithm. Another way of viewing this problem is through the lens of energy efficiency. Each channel access—whether for sending or listening—consumes energy. Most work on contention resolution is ***sending efficient***, but is not ***listening efficient*** (e.g., [7, 10, 11, 24, 25, 36, 55–57]). That is, most protocols optimize how frequently a packet sends, but allow a packet to listen in every slot "for free".

In fact, *both* sending and listening are expensive operations (e.g., [26, 43, 54, 67]), and minimizing energy usage by having devices sleep as much as possible has been a long-standing and popular strategy to maximize the network lifetime (for example, the development of duty-cycle protocols [44, 47, 68]). Why optimize listening in contention resolution protocols if packets must receive messages (for other network applications)? Informally, there are two types of listening: listening to receive messages, and listening to execute a contention resolution protocol (which is what enables sending messages). To receive messages, packets do not need to listen in every time slot; there exist methods for optimizing this first type of listening. Although this is beyond the scope of this paper, as one example, in many wireless settings, a central base station can monitor the channel constantly and facilitate message exchange [27].

We call a protocol ***fully-energy efficient*** if it is both sending efficient and listening efficient. By definition, such a protocol cannot have short feedback loops, since it can access the channel only rarely.

Past work: Minimizing listening by allowing for explicit synchronization. Currently, the only known path to full energy efficiency is via *explicit synchronization*. This means that the model is extended so that packets can send synchronization messages to each other whenever they broadcast [12, 25, 46]. In [12], these synchronization messages have size $\Theta(\log N)$ bits each, which means that an arbitrary polynomial amount of communication can be expressed in a slot. In [25, 46], the synchronization messages are smaller ($O(1)$ bits), but some packets are permitted to stick around as "Good Samaritans" in order to serve as long-term coordinators (that send many messages via many broadcasts).

Using $\Theta(\log N)$-bit synchronization messages, Bender, Kopelowitz, Pettie, and Young [12] give an algorithm with $\Theta(1)$ throughput and expected $O(\log(\log^* N))$ channel accesses per packet. Using Good-Samaritan packets, De Marco and Stachowiak [25] and De Marco, Kowalski, and Stachowiak [46] provide a constant-throughput algorithm that is *sending* efficient ($O(\log N)$ transmissions per packet) and conjecture that their techniques can be extended to achieve fully energy efficiency ($O(\text{polylog}\, N)$ channel accesses per packet), with high probability and even without collision detection.

In all of these cases, even with the help of explicit synchronization, it remains open whether one can achieve such results in the presence of adversarial noise. Indeed, noise has the potential to be extra-problematic for energy-efficient algorithms since these algorithms listen to the channel less frequently and can potentially be thrown off by a small amount of well-placed noise.

This paper. We show that it is indeed possible to achieve constant throughput with full energy efficiency while being robust to adversarial noise. Moreover, these results hold in the standard ternary-feedback model, without requiring the addition of any sort of explicit synchronization.

Our algorithm belongs to a natural family of multiplicative-weight-update algorithms (e.g., [7, 14, 52, 55–57]). When a packet hears silence, it multiplicatively increases both its listening and sending probabilities. Conversely, when a packet hears noise, it multiplicatively decreases these probabilities. There are no control messages, no Good Samaritan packets, and no leaders elected.

What makes our algorithm/analysis interesting is that we are able to support a multiplicative-weight-update framework while having each packet 'cover its eyes' almost all of the time. This is in stark contrast to prior constant-throughput algorithms, which adjust sending probabilities in every slot. Because each packet listens to so few slots, the different packets that are in the system at the same time may end up with very different perspectives on the world from each other. In order to analyze the 'herd behavior' of the packets in this potentially chaotic setting, substantially new techniques end up being required. These techniques are also what allow us to handle the additional chaos that adversarial jamming adds to the system.

1.1 Model

A *finite* or *infinite* stream of indistinguishable packets arrives over time; the number of arrivals is unknown to the algorithm. Each packet must be sent on the ***multiple-access channel***. Time is divided into synchronized slots, each of which is sufficiently large to send a single packet. An ***adversary*** (specified below) controls how many packets are injected into the system in each slot. When a packet successfully transmits, it departs the system. There is no universal numbering scheme for the slots; that is, there is no global clock from which a packet could infer the system lifetime or slot parity. Additionally, the packets do not receive any additional information about how many packets have arrived or will arrive. Instead, packets only receive information through the ternary feedback model.

We now describe the ***ternary feedback model***. Initially, we will define the model without jamming. In each time slot, each packet in the system can take one of three actions: (i) sleep, (ii) send, or (iii) listen to the channel. Packets that take actions (ii) or (iii) are said to ***access the channel***. If no packets choose to send during a slot, then that (non-jammed) slot is ***empty***/***silent***; if exactly one packet sends, then that (non-jammed) slot is ***full*** and ***successful***; if two or more packets send, then that slot is ***full*** and ***noisy***. A packet that listens during a slot (action iii) learns whether the slot was (0) empty, (1) successful, or (2+) noisy. A packet that sleeps during a slot (action i) learns nothing about the state of the slot. A packet that sends (action ii), either ***transmits successfully*** and leaves the system, or ***collides*** and remains in the system.[2]

We now add jamming to the picture; an adversary determines which slots are jammed. To jam a particular slot, the adversary broadcasts ***noise*** into that slot. All jammed slots are thus ***full*** and ***noisy***. A packet that listens in a jammed slot hears that the slot was noisy, but does not know whether that noise came from jamming or was merely a collision between two or more packets. A packet that sends during a jammed slot ***collides*** and thus remains in the system.

An ***adversary*** determines, for each slot t, how many packets to inject in slot t and whether to jam in that slot. This paper considers an ***adaptive adversary***, which bases its decision on the entire state of the system so far, i.e., up to the end of slot $t-1$, but not the outcomes of future coin tosses. Thus, if at slot t, a packet p decides whether to send based on a coin flip, the adaptive adversary does *not* get to see that coin flip until after slot t.

A basic metric: (overall) throughput. The main objective of contention resolution is to optimize ***throughput***, defined next. A slot is ***active*** if at least one packet is in the system during that slot; ***inactive*** slots can be ignored in our analysis. Without loss of generality, assume throughout that the first slot is active. Without jamming, the ***throughput at time t*** is defined as T_t/S_t, where $\boldsymbol{T_t}$ is the number of successful transmissions during slots $1, 2, \ldots, t$, and $\boldsymbol{S_t}$ is the number of active slots in the same time interval.[3] On a finite input, the ***(overall) throughput*** is defined with respect to the final active slot t; at this point, the overall throughput is N/S, where $N = T_t$ is the total number of packets and $S = S_t$ is the total number of active slots. Overall throughput is not well-defined on an infinite execution.

[2] For ease of presentation, in our algorithms, we say that a packet can listen and send simultaneously, but any packet that is sending actually does not need to listen to determine the state of the channel. If the packet is still in the system after sending in slot t, then slot t was noisy.

[3] By assumption that the first slot is active, we have $S_t \geq 1$ and hence throughput is always well defined.

A deficiency of the throughput metric (when defined naïvely) is that even if an algorithm guarantees $\Theta(1)$ overall throughput, it is not possible to achieve $\Theta(1)$ throughput uniformly across time. For example, if there is a burst of N packets at time 0, and N is unknown to the algorithm, then the throughput will be 0 for a superconstant number of slots (e.g., see [15, 64]), and this is *provably unavoidable* regardless of the backoff strategy being used. Thus (overall) throughput is only meaningful at the end of the execution, or at points in time where there are no packets in the system.

A stronger metric: implicit throughput [11]. In order to support a meaningful notion of throughput, even at intermediate points in time, Bender, Kopelowitz, Kuszmaul, and Pettie [11] propose a refined definition that they term "implicit throughput" [11]. The ***implicit throughput*** at time t is defined as N_t/S_t, where N_t is the total number of packets that arrive at or before time t and S_t is the total number of active slots so far.[4]

One perspective on implicit throughput is that it is an *analytical tool*. Indeed, whenever we reach a point in time where overall throughput is meaningful (i.e., there are no packets left in the system), the two metrics become provably equal. This includes both at the end of any finite execution or during quiet periods of infinite executions. Another perspective on implicit throughput is that it is a stronger metric that offers a meaningful guarantee even at intermediate points in time: what constant implicit throughput means is that the number of active slots used so far should never be asymptotically larger than the number of packets that have arrived.

Throughout the rest of the paper, we focus exclusively on implicit throughput. We should emphasize, however, that this only makes our results stronger—the results also imply the standard constant-throughput guarantees that one would normally strive for.

Extending to adversarial jamming. We next extend the definitions of throughput and implicit throughput for the case of jamming following [10]. An algorithm wastes a slot if that slot has silence or a collision, and throughput measures the fraction of slots that the algorithm could have used but instead wasted. Let $\mathcal{J}_t$ denote the number of jammed slots through slot t. Then, the throughput of an execution ending at time t is defined to be $(T_t + \mathcal{J}_t)/S_t$, and the implicit throughput at slot t is defined to be $(N_t + \mathcal{J}_t)/S_t$.

Some useful properties of implicit throughput, and applications to adversarial queuing theory. We conclude the section by summarizing several useful properties of implicit throughput.

Observation 1 ([11]). *Consider any inactive slot t, i.e., where there are no active packets in the system. Then the implicit throughput and throughput are the same at slot t.*

Observation 2 ([11]). *Let δ be any lower bound on the implicit throughput of an algorithm; that is, suppose the algorithm achieves implicit throughput of at least δ at all times. Let N_t and S_t denote the total number of packet arrivals and active slots, respectively, at or before time t. Then $S_t \le N_t/\delta$. Consequently:*

- ***Overall throughput.*** *Suppose that there are a total of $N \ge 1$ packet arrivals. Then the total number of active slots is at most N/δ, and hence the overall throughput is at least δ.*
- ***Backlog reprieve.*** *If $N_t < \delta t$, then there exists an inactive slot $t' \le t$. Thus, all packets that arrived before t' have completed before t', and hence the throughput at time t' is at least δ.*

Finally, we observe that there are several natural settings in which implicit-throughput guarantees directly imply strong guarantees on packet backlog, even for infinite input sequences. Suppose, in particular, that packets arrive according to adversarial queuing theory, which parameterizes the "burstiness" of packet-arrival in infinite streams. In adversarial queuing theory [8, 13, 19, 22], the adversary is restricted from injecting too many packets/jammed slots over a set of consecutive slots of length S, where S is a parameter of the model that we refer to as the ***granularity***. For granularity S, the number of packet arrivals plus the number of jammed slots is limited to λS, where the ***arrival rate*** λ is a constant less than one. On the other hand, how the packet arrivals are distributed within each S-sized window is adversarial: no restrictions are placed on how the (at most) λS packets and jammed slots are distributed. By showing that the implicit throughput of all active slots is $\Omega(1)$, we obtain as a corollary a strong bound of $O(S)$ on the number of packets backlogged in the system at all times, as long as λ is a sufficiently small constant.

1.2 Main results

We present an algorithm that with high-probability guarantees $\Omega(1)$ implicit throughput in all active slots and full energy efficiency. Thus, we resolve two open questions in contention resolution: First, we show that full-energy efficiency is feasible, even with only ternary feedback. Second, we show that these guarantees are achievable, even in the presence of adversarial jamming.

Our algorithm relies on the natural multiplicative-weight approach to backoff—with a careful choice of probabilities and updates rules. In contrast to some previous approaches, we do not rely on packet batching (to turn the online problem into a series of batch problems), leader election, busy tones, population estimation, dividing the channel into simulated subchannels (e.g., odd and even slots) for packet coordination, or other approaches seen in many modern algorithms; alas, these seem problematic in our setting. Given the simplicity of our algorithm, the technical innovation lies in choosing the update parameters, analyzing the underlying combinatorial process, and proving that it is fast, robust, and fully energy efficient.

Our algorithm Low-Sensing Backoff guarantees the main theorems below.[5]

Theorem 3 (Implicit throughput, infinite packet streams). *At the t-th active slot, the implicit throughput is $\Omega(1)$ w.h.p. in t.*

Corollary 4 (Throughput, finite packet streams). *Consider an input stream of N packets with $\mathcal{J}$ jammed slots. The throughput for the execution is $\Theta(1)$ w.h.p. in $N + \mathcal{J}$.*

Corollary 5 (Bounded backlog for adversarial-queuing arrivals). *Consider adversarial-queuing-theory arrivals with a*

[4] Notice that while N_t depends on the adversary, the number of active slots depends on the algorithm, whose goal it is to make slots inactive by completing packets.

[5] An event occurs ***with high probability (w.h.p.) in x*** if for any fixed constant $c \ge 1$, the probability of the event is at least $1 - x^{-c}$.

sufficiently small constant arrival-rate λ and granularity S; i.e., in any interval of length S, the total number of packet arrivals and jammed slots is at most λS. Then, for any given slot t, the number of packets currently in the system is at most $O(S)$ w.h.p. in S.

Theorem 6 (Energy, finite executions). *Consider a finite execution with N total packet arrivals and $\mathcal{J}$ total jammed slots against an adaptive adversary. Any given packet accesses the channel $O(\text{polylog}(N+\mathcal{J}))$ times w.h.p. in $N+\mathcal{J}$.*

Theorem 7 (Energy, adversarial queuing). *Consider a (finite or infinite) packet stream with adversarial-queuing arrivals with granularity S and where the arrival rate is a sufficiently small constant. Then any given packet accesses the channel $O(\text{polylog}(S))$ times w.h.p. in S against an adaptive adversary.*

Theorem 8 (Energy, infinite executions). *Consider an infinite packet stream, and let N_t and $\mathcal{J}_t$ denote the number of arrivals and jammed slots, respectively, up until time t against an adaptive adversary. Then any given packet accesses the channel $O(\text{polylog}(N_t+\mathcal{J}_t))$ times before time t, w.h.p. in $N_t+\mathcal{J}_t$.*

Theorem 3 is proved in Section 5.5 where it appears as Corollary 30. Corollary 5 corresponds to Corollary 33. The remaining theorems are proved in Section 5.6. In particular, Theorem 6 corresponds to Theorem 34, and Theorem 7 corresponds to Theorem 36. Finally, Theorem 8 is included in Theorem 38. The details of these results are presented in the full version of our paper [9].

1.3 Extensions to Reactive Adversary

A ***reactive adversary*** [40, 56, 63] has an instantaneous reaction time; that is, this adversary listens to the channel and can decide whether to jam and/or inject new packet(s) in slot t based on what it hears in slot t itself. In contrast, the standard adaptive adversary would not know whether any packet chooses to send in slot t until slot $t+1$. This allows a reactive adversary to cheaply prevent any particular packet p from transmitting successfully by jamming only those slots where p makes transmission attempts. Thus, against a reactive adversary, the total number of channel accesses required is at least linear in the amount of jamming. For exponential backoff, the situation is more dire: for any T a reactive adversary can also drive the throughput down to $O(1/T)$ by jamming a single packet a mere $\Theta(\log T)$ times.

Note that reactivity and adaptivity are somewhat orthogonal. Reactivity addresses how quickly the adversary can react to the detectable channel state—importantly, only sending is revealed, since it is detectable. In contrast, an adaptive adversary knows all of the internal state and random choices of packets up to the previous slot, and in particular this adversary also knows if packets choose to listen.

In addition to our main results on purely adaptive adversaries, we also address an adversary that is *both* adaptive and reactive. It turns out that the reactive adversary does not impact our implicit-throughput bounds for our algorithm (our analysis applies whether or not the adversary can see the channel activity at the current time). Reactivity thus only impacts the number of channel accesses. Roughly speaking, the theorem states that the reactive adversary has nontrivial impact on the worst-case number of channel accesses and thus energy, which is to be expected, but it does not have significant impact on the average.

Low-Sensing Backoff for packet u
Key Variables:
- $w_u(t)$: window size of u in slot t.
  If u is injected at time slot t, then $w_u(t) = w_{\min}$.
- c: a sufficiently large positive constant.

In every slot t, packet u executes the following four steps with probability $\frac{c\ln^3(w_u(t))}{w_u(t)}$:
- **Listen**
- **Send** with probability $\frac{1}{c\ln^3(w_u(t))}$
- If u heard a silent slot, then
  $w_u(t+1) \leftarrow \max\left\{\frac{w_u(t)}{1+1/(c\ln(w_u(t)))},\ w_{\min}\right\}$
- If u heard a noisy slot, then
  $w_u(t+1) \leftarrow w_u(t)\cdot\left(1+\frac{1}{c\ln(w_u(t))}\right)$

Figure 1: Low-Sensing Backoff algorithm.

Theorem 9 (Energy, reactive adversary). *The following apply to a reactive and adaptive adversary.*

(1) **Finite streams.** *Consider a finite execution with N total packet arrivals and $\mathcal{J}$ total jammed slots. Any given packet accesses the channel $O((\mathcal{J}+1)\,\text{polylog}(N))$ times w.h.p. in $N+\mathcal{J}$. Moreover, the average number of channel accesses is only $O((\mathcal{J}/N+1)\,\text{polylog}(N+\mathcal{J}))$ times w.h.p. in $N+\mathcal{J}$.*

(2) **Adversarial queuing.** *Consider a (finite or infinite) packet stream with adversarial-queuing arrivals with granularity S and where the arrival rate is a sufficiently small constant. Then any given packet accesses the channel at most $O(S)$ times, w.h.p. in S. In addition, the average number of accesses per slot is $O(\text{polylog}(S))$, w.h.p. in S.*

(3) **Infinite executions.** *Consider an infinite packet stream, and let N_t and $\mathcal{J}_t$ denote the number of arrivals and jammed slots, respectively, up until time t. Then any given packet accesses the channel $O((\mathcal{J}_t+1)\,\text{polylog}(N_t+\mathcal{J}_t))$ times before time t, w.h.p. in $N_t+\mathcal{J}_t$. Moreover, the average number of channel accesses is $O((\mathcal{J}_t/N_t+1)\,\text{polylog}(N_t+\mathcal{J}_t))$.*

The items of Theorem 9 are each proved separately as Theorems 35, 37, and 38 in Section 5.6 of our full paper [9].

2 LOW-SENSING BACKOFF ALGORITHM

This section presents the Low-Sensing Backoff algorithm; see Figure 1. For ease of presentation, we describe our algorithm as listening whenever it sends. However, the packet need not actually do both; observe that any packet that is sending does not need to listen to determine the state of the channel, since if the packet is still in the system after sending in slot t, then slot t was noisy.

The probabilities for sending and listening in Low-Sensing Backoff are determined by a single parameter, which we call

packet u's window size. Let $w_u(t)$ denote packet u's window size at time slot t. When u is injected into the system, its window size is set to the minimum allowed value: $\boldsymbol{w_{\min}} \geq 2$. Let c be a sufficiently large positive constant. The sending and listening rules are as follows. First, packet u listens with probability $c \ln^3(w_u(t))/w_u(t)$. Then, conditioned on listening, u sends with probability $1/(c \ln^3(w_u(t)))$.

A packet u only has the option to change its window size when it accesses the channel. Specifically, if at time t, packet u listens to the channel and learns that the slot t is busy, then the window size increases (or *backs off*) by a ***backoff factor*** of $1 + 1/(c \ln(w_u(t)))$; that is, $w_u(t+1) \leftarrow w_u(t)(1 + 1/(c \ln(w_u(t))))$. Similarly, if at time t, packet u accesses the channel and learns that the slot t is empty, then the window size shrinks (or *backs on*) by a ***backon factor*** of $1 + 1/c \ln(w_u(t))$, or until it gets back down to $w_{\min}$, that is, $w_u(t+1) \leftarrow \max\{w_u(t)/(1 + 1/(c \ln w_u(t))), w_{\min}\}$.

3 TECHNICAL OVERVIEW

This section gives a technical overview. In Section 3.1, we introduce the notion of contention. In Section 3.2, we introduce our potential function $\Phi(t)$. Section 3.3 gives the main structure of our analysis in terms of intervals. Finally, Section 3.4 provides a synopsis of the main analytical results achieved and how they are deployed to make progress towards our main results (Section 1.2). That is, we describe the main point of each of the technical sections; namely, Sections 5.1–5.6, which are available in the full version of our paper [9].

3.1 Contention

For any slot t, we define the ***contention*** $\boldsymbol{C(t)} = \sum_u 1/w_u(t)$ to be the sum of the sending probabilities in that slot, i.e., the expected number of packets that attempt to send during that slot. We say contention is ***high*** when $C(t) > C_{\text{high}}$, where $\boldsymbol{C_{\text{high}}} > 1$ is some fixed positive constant. Conversely, we say that contention is ***low*** when $C(t) < C_{\text{low}}$, where we define $\boldsymbol{C_{\text{low}}}$ to be some fixed positive constant such that $C_{\text{low}} \leq 1/w_{\min}$. Otherwise, if contention is in $[C_{\text{low}}, C_{\text{high}}]$, then we say that contention is ***good***.

3.2 Our potential function

Throughout the execution of Low-Sensing Backoff, we maintain a potential function $\Phi(t)$ that captures the state of the system at time t and measures the progress toward delivering all packets. When a slot t is inactive, $\Phi(t) = 0$. We will see that packet arrivals increase $\Phi(t)$ by $\Theta(1)$ per newly arrived packet, that packets exiting the system successfully decrease $\Phi(t)$ by $\Theta(1)$ per packet, that a jammed slot increases $\Phi(t)$ by $O(1)$, and that on average each slot decreases the potential by $\Theta(1)$, ignoring newly arrived packets.

For any slot t, $\boldsymbol{N(t)}$ is the number of packets in the system, $\boldsymbol{w_u(t)}$ is u's window size, $\boldsymbol{w_{\max}(t)}$ is the largest window size over all packets, and $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, and $\boldsymbol{\alpha_3}$ are positive constants. Our potential function consists of three terms. Implicitly, the third term is 0 if there are no packets in the system (and thus $w_{\max}(t) = 0$):

$$\Phi(t) = \alpha_1 N(t) + \alpha_2 \sum_u \frac{1}{\ln(w_u(t))} + \alpha_3 \frac{w_{\max}(t)}{\ln^2(w_{\max}(t))}.$$

We abbreviate $\Phi(t)$ as:

$$\Phi(t) = \alpha_1 N(t) + \alpha_2 H(t) + \alpha_3 L(t),$$

where α_1, α_2, and α_3 may be set so that $\Phi(t)$ will decrease as time progresses for all values of contention $C(t)$. The notation $\boldsymbol{H(t)}$ and $\boldsymbol{L(t)}$ is used to highlight that these terms capture the impact on Φ from high contention and low contention, respectively.

Why these terms? There are three main features of the state of the system that are captured by the potential function: the number of packets, the contention, and the size of the windows. (Note that these are not independent, as larger windows correspond to lower contention.) Roughly speaking, when there are many active packets, potential should be high, and when there are no packets, the potential should be 0. The $N(t)$ term captures this idea directly by counting the number of packets.

The $H(t)$ term is chosen so that the expected change to $H(t)$ in a slot is proportional to the contention. When the contention is high (and the slot is most likely to have a collision), in expectation $H(t)$ decreases proportional to the contention (due to the update rule on noisy slots). On the other hand, when the contention is low (and the slot is most likely to have silence), $H(t)$ increases proportional to the contention. Overall, this is pretty great: when contention is high, $H(t)$ is likely to decrease by a lot. When contention is low, there is a small expected increase, but that increase is counterbalanced by the (small) expected number of packet successes reflected in $N(t)$. Choosing $\alpha_1 > \alpha_2$ makes the net effect a decrease.

Finally, the $L(t)$ term allows us to cope with the situation that the contention is low but some packets in the system have large windows (e.g., there is a single packet with a very large window). As it is likely to take a long time for the packet to succeed, the potential should be high. $L(t)$ is roughly the expected time for a packet with window $w_{\max}(t)$ to decrease its window size to a constant if all slots are silent. The analysis then needs to show that any increases to $L(t)$ are counterbalanced by decreases in the other terms, ensured by $\alpha_1 > \alpha_2 > \alpha_3$.

Challenge with $L(t)$. The $N(t)$ and $H(t)$ terms are well-behaved in the sense that they change on a *per-slot basis*, while the $L(t)$ term cannot. To see why, consider the case that several packets with window size $w_{\max}(t)$ remain. The $L(t)$ term only decreases after *all* of those packets have chosen to listen and observed silence. On any step that multiple such packets remain, it is extremely unlikely that all of the packets choose to listen. Thus, $L(t)$ does not decrease by a constant in expectation. Instead, we need a coarser granularity to understand the behavior of $L(t)$.

3.3 Analyzing intervals

Our analysis divides the execution into disjoint ***intervals*** of time. The first interval starts at the first step with an active packet. An interval starting at time t has size $\tau = (1/c_\tau)\max\{L(t), \sqrt{N(t)}\}$, where c_τ is a constant. If any active packets remain, the next interval starts immediately after the previous interval ends. (Otherwise, an interval begins the next time there is an active packet.)

A key technical theorem is the following. Let $\mathcal{A}$ and $\mathcal{J}$ denote the number of arrivals and jammed slots, respectively, in the size-τ interval. For $\mathcal{A} = \mathcal{J} = 0$, the lemma states that the potential decreases by $\Omega(\tau)$ across the interval, with high probability in τ,

meaning a decrease of $\Omega(1)$ per slot. For general $\mathcal{A}, \mathcal{J}$, the potential decreases by $\Omega(\tau) - O(\mathcal{A} + \mathcal{J})$.

Theorem 27 (Decrease in $\Phi(t)$ over interval $\mathcal{I}$ w.h.p. in $|\mathcal{I}|$). *Consider an interval $\mathcal{I}$ starting at t of length $|\mathcal{I}| = \tau = (1/c_\tau) \cdot \max\left\{\frac{w_{\max}(t)}{\ln^2(w_{\max}(t))}, N(t)^{1/2}\right\}$. Let $\mathcal{A}$ and $\mathcal{J}$ be the number of packet arrivals and jammed slots in $\mathcal{I}$. With high probability in τ, Φ decreases over $\mathcal{I}$ by at least $\Omega(\tau) - O(\mathcal{A} + \mathcal{J})$. That is,*

$$\Pr\left[(\Phi(t') - \Phi(t)) \;\geq\; \Theta(\mathcal{A} + \mathcal{J}) - \Omega(\tau)\right] \;\leq\; (1/\tau)^{\Theta(1)} \,.$$

Our proof of Theorem 27 is broken into several lemmas according to the level of contention. Specifically, we have separate cases for high contention, good contention, and low contention. In each of the cases, absent arrivals and jamming, we argue that there is a net decrease in potential, with high probability, but the contributing term is different in each case. The interplay between $N(t)$ and $H(t)$ is tight enough that we analyze the net effect on the sum of these terms together, but we analyze $L(t)$ separately. A more detailed summary is provided next in Section 3.4.

A significant complication is that (1) the probability stated in Theorem 27 depends on the size of the interval, and (2) the interval sizes are determined adaptively by actions of the adversary. To analyze the full process, we model an execution as a specific biased random walk that we set up as a betting game (Section 5.5). The bounds provided by the betting game translate into high-probability bounds with respect to the total number of packets.

Throughout the paper, standard Chernoff bounds sometimes cannot be used for two reasons. First the adversary can adaptively influence the length of an interval. Moreover within each interval, the adversary can influence which slots are high, low, and good contention. To be able to analyze these slots separately, we must instead apply a generalization of Azuma's inequality (see Theorems 13 and 14 in Section 5.1, taken from [42]) that gives us Chernoff-like bounds but with adaptively chosen probability distributions.

Finally, good upper bounds for $\Phi(t)$ enable us to characterize the (implicit and standard) throughput and energy consumption of Low-Sensing Backoff in all its variety of settings (finite versus infinite executions, arbitrary infinite versus infinite with adversarial-queuing arrivals, adaptive adversaries that are reactive versus non-reactive). The most direct application of $\Phi(t)$ is to bound implicit throughput. $\Phi(t)$ also gives us an upper bound on the maximum window size $w_{\max}(t)$, specifically, $w_{\max}(t) = O\big(\Phi(t)\log^2(\Phi(t))\big)$, which we use to prove energy bounds in Section 5.6.

3.4 Proof organization

The main analysis in this paper, including all of the proofs, appears in Section 5 in our Appendix, which is available in the full version of our paper [9]. This section summarizes the proof structure, highlighting the key lemma statements.

Overview of Section 5.1. Preliminaries. This section lists several well-known inequalities that are used throughout our analysis. We review bounds on the probability that a slot is noisy, empty, or contains a successful transmission as a function of contention (Lemmas 10, 11 and 12). The lemmas in this section allow us to immediately obtain constant bounds on the probabilities of empty slots, successful slots, and noisy slots in different contention regimes.

Theorems 13 and 14 give upper and lower bounds for the sum of random variables, where the distribution of each subsequent random variable is determined by an adaptive adversary. This adversarial, multiplicative version of Azuma's inequality is a powerful tool from [42] that allows us to analyze the performance of our algorithm in situations where a simpler Chernoff-bound-style argument does not appear to work, given the adaptive nature of our adversary.

Overview of Section 5.2. $N(t)+H(t)$: over single slots and intervals, when contention is low, high, and good. This section addresses the behavior of $N(t)$ and $H(t)$. When contention is high, we expect to see a decrease in $H(t)$, which should be large enough that its reduction outweighs any increase from $L(t)$, and thus $\Phi(t)$ decreases.

Lemma 18 shows how much $H(t)$ changes as a result of a *specific* packet listening during a slot t—that is, how much $H(t)$ increases when the slot is silent and decreases when the slot is noisy. Lemma 19 analyzes the change to $N(t) + H(t)$ due to the low and good contention slots in an arbitrary interval. We highlight that the adaptive adversary exerts some control over which slots have low and good contention, since it can inject packets and/or jam in slot $t+1$ based on the packets' random choices in slot t. In particular, let $|\mathcal{G}|$ denote the number of good slots in the interval, then Lemma 19 shows the following. Over *good-contention* slots, $N(t) + H(t)$ decreases by $\Omega(|\mathcal{G}|)$, minus the number of packet injections, jammed slots, and a polylog term in the length of the interval, w.h.p. in the interval length. Lemma 19 also shows that over the *low-contention slots*, $N(t) + H(t)$ increases by at most the number of packet injections, jammed slots, and a polylog term in the length of the interval, again w.h.p. in the interval length.

Lemma 18 (Increase/decrease in $H(t)$ due to a silent/noisy slot). *When packet u listens to a silent slot t, $H(t)$ increases by $\Theta\big(\frac{1}{c\ln^3 w_u}\big)$ due to packet u. When a packet u listens to a noisy slot, $H(t)$ decreases by $\Theta\big(\frac{1}{c\ln^3 w_u}\big)$ due to packet u.*

Lemma 19 (Net delta in contribution of $N(t)$ and $H(t)$ to potential over low and good contention slots). *Let $\mathcal{I}$ be an arbitrary interval starting at time t with length $|\mathcal{I}| = \tau$. Let $\mathcal{L}$ be the set of slots in $\mathcal{I}$ during which $C(t) \leq C_{\text{low}}$. Let $\mathcal{G}$ be the set of slots in $\mathcal{I}$ during which $C(t) > C_{\text{low}}$ and $C(t) \leq C_{\text{high}}$. Let $\mathcal{A}_{\mathcal{L}}$ be the number of packet arrivals in time slots in $\mathcal{L}$. Let $\mathcal{A}_{\mathcal{G}}$ be the number of packet arrivals in time slots in $\mathcal{G}$. Let $\mathcal{J}_{\mathcal{L}}$ be the number of jammed slots in $\mathcal{L}$. Let $\mathcal{J}_{\mathcal{G}}$ be the number of jammed slots in $\mathcal{G}$. Define:*

- *the* **net delta** *over $\mathcal{L}$ to be the sum of the changes in $N(t)$ and $H(t)$ during the slots in $\mathcal{L}$, i.e.,*

$$\sum_{t' \in \mathcal{L}} \Big(\alpha_1\big(N(t'+1) - N(t')\big) + \alpha_2\big(H(t'+1) - H(t')\big)\Big).$$

- *the* **net delta** *over $\mathcal{G}$ to be the sum of the changes in $N(t)$ and $H(t)$ during the slots in $\mathcal{G}$, i.e.,*

$$\sum_{t' \in \mathcal{G}} \Big(\alpha_1\big(N(t'+1) - N(t')\big) + \alpha_2\big(H(t'+1) - H(t')\big)\Big).$$

Then, for proper choices of α_1 and α_2:

- *The net delta over $\mathcal{L}$ is at most $O(\ln^2 \tau) + \alpha_1(\mathcal{A}_{\mathcal{L}} + \mathcal{J}_{\mathcal{L}})$ w.h.p. in τ.*

- *The net delta over $\mathcal{G}$ is at most $O(\ln^2 \tau) + \alpha_1(\mathcal{A}_\mathcal{G} + \mathcal{J}_\mathcal{G}) - \Omega(|\mathcal{G}|)$ w.h.p. in τ.*

Lemma 20 provides a symmetric high-probability bound on $N(t) + H(t)$ over the *high-contention* slots in an arbitrary interval. For the high-probability bound, in this case, we have that $N(t) + H(t)$ will decrease by $\Omega(|\mathcal{H}|)$, where $|\mathcal{H}|$ is the number of high-contention slots, up to the usual additional terms of jamming, packet injections, and a polylog term in terms of the interval length.

Lemma 20 (Net delta in contribution of $N(t)$ and $H(t)$ to potential over high contention slots). *Let $\mathcal{I}$ be an arbitrary interval starting at time t with length $|I| = \tau$. Let $\mathcal{H}$ be the set of slots in $\mathcal{I}$ during which $C(t) > C_{\text{high}}$. Let $\mathcal{A}_\mathcal{H}$ be the number of packet arrivals in time slots in $\mathcal{H}$, and let $\mathcal{J}_\mathcal{H}$ be the number of jammed slots in $\mathcal{H}$,*

Define the **net delta** *over $\mathcal{H}$ to be the sum of the changes in N and H during the slots in $\mathcal{H}$, i.e.,*

$$\sum_{t' \in \mathcal{H}} \Big(\alpha_1\big(N(t'+1) - N(t')\big) + \alpha_2\big(H(t'+1) - H(t')\big)\Big).$$

Then the net delta over $\mathcal{H}$ is at most $O(\ln^3 \tau) + \alpha_1 \mathcal{A}_\mathcal{H} - \Omega(|\mathcal{H}|)$ w.h.p. in τ.

It is worth noting that the proofs of Lemmas 19 and 20 are technically involved. One of the reasons for this is that these lemmas contain our main applications of Theorems 13 and 14. This is necessary because the potential-function terms behave very differently in the three contention regimes—and because the adaptive adversary has the ability to change the contention in a slot on the fly.

Lemma 21 then collates Lemmas 19 and 20 to show that over an arbitrary interval of length τ, it is either the case that almost all of the slots are low contention slots, or the first two terms decrease by $\Omega(\tau)$ with high probability (again, up to terms for packet insertions and jamming). Lemma 21 considers all slots, rather than only those of a particular contention regime. This lemma is the only one from this subsection that will be used later in the analysis, but the earlier lemmas in the subsection are necessary to build up to it.

Lemma 21 (Unless most slots have low contention, $\alpha_1 N(t) + \alpha_2 H(t)$ decreases). *Let $\mathcal{I}$ be an arbitrary interval of length $\tau > \Omega(1)$ with $\mathcal{A}$ packet arrivals and $\mathcal{J}$ jammed slots. With high probability in τ, at least one of the following two conditions holds:*

- *Less than $1/10$ of slots satisfy $C(t) \geq C_{\text{low}}$.*
- *$\alpha_1 N(t) + \alpha_2 H(t)$ decreases by $\Omega(\tau) - O(\mathcal{A} + \mathcal{J})$ over $\mathcal{I}$.*

Additionally, $\alpha_1 N(t) + \alpha_2 H(t)$ increases by at most $O(\ln^3 \tau + \mathcal{A} + \mathcal{J})$ w.h.p. in τ.

Overview of Section 5.3. Amortized behavior of $L(t)$. This section analyzes $L(t)$'s behavior over intervals of length $|\mathcal{I}| = \tau = (1/c_\tau) \cdot \max\{\frac{w_{\max}(t)}{\ln^2(w_{\max}(t))}, N(t)^{1/2}\}$. The two main things that we want to show are that $L(t)$ does not increase by much, regardless of the contention regime, and that when there are many low-contention slots, $L(t)$ exhibits a substantial decrease.

Lemma 24 argues that a packet with large-enough window size is unlikely to have its window change by much during the interval. This lemma is instrumental when considering packets across an interval (notably in the proof of Lemma 26) as it means that their probability of listening also does not change by much.

Lemma 25 provides one of the main results of the section: a tail bound, and hence also a high probability bound, on how much L increases over the interval regardless of contention. The proofs for both Lemmas 24 and 25 amount to arguing that an individual packet is unlikely to listen to the channel too many times, which means that its window size also cannot change by very much. Because we are pessimistically counting the number of listens, the actual state of the channel does not appear in the proofs, and thus the number of jammed slots is irrelevant.

Lemma 24 (Bounds on the factor that a large window can grow/shrink). *Consider any packet during an interval $\mathcal{I}$ with $\tau = |\mathcal{I}|$. Let Z satisfy $Z/\ln^2(Z) = \tau$. And let $W \geq w_{\min}$ be the initial size of the packet's window. Let W^- be the smallest window size the packet has while still active in the interval, and let W^+ be the biggest window size the packet achieves during the interval. Then for large enough choice of constants $w_{\min}$ and c and any constant parameter $\gamma > 0$ and $k \geq 2$:*
If $W = \Theta(kZ)$, then

$$\Pr\left[W^+ \geq e^\gamma W \text{ or } W^- < W/e^\gamma\right] \leq 1/\tau^{\Theta(c\gamma \lg(\gamma k))}.$$

Lemma 25 (Tail bound on increase in $L(t)$). *Consider an interval $\mathcal{I}$ with length $\tau = |\mathcal{I}|$ starting from time t and ending at time $t' = t + \tau$. Let $\mathcal{A}$ be the number arrivals during the interval. Then for for large-enough constant c in the algorithm and any $k \geq 2$:*

$$\Pr\left[L(t') \geq \Theta(\mathcal{A} + k\tau)\right] \leq 2^{-\Theta(c(\lg\tau \cdot \lg k + \lg^2 k))}.$$

Lemma 26 is the other main result of the section. This lemma says that as long as most slots have low contention, then L decreases by $\Omega(\tau)$, minus the number of packet arrivals and jammed slots. The proof focuses on packets with large windows, i.e., window closes to $w_{\max}(t)$. The main idea of the proof is to give a high-probability lower bound on the number of times each such packet listens and hears silence as well as an upper bound on how many times the packet listens and hears noise. As long as the former is larger by a constant factor, the packet is likely to decreases its window size by a constant factor. Taking a union bound across packets is enough to conclude that all packets with large windows have their window sizes decrease, with high probability.

Lemma 26 (Mostly low contention implies decrease in $L(t)$). *Consider an interval $\mathcal{I}$ starting at t of length τ, where $\tau = (1/c_\tau) \max\left\{L(t), \sqrt{N(t)}\right\}$. Let $t_1 = t + \tau$, and let $\mathcal{A}$ and $\mathcal{J}$ denote the number of packet arrivals and jammed slots, respectively, over $\mathcal{I}$. Then, with high probability in τ, either*

- *$L(t_1) \leq L(t)/d + O(\mathcal{A})$, where $d > 1$ is a constant, or*
- *At least a $1/10$-fraction of the slots t' in the interval $\mathcal{I}$ are either jammed or have contention $C(t') \geq C_{\text{low}}$.*

Incorporating the fact that $\tau \geq L(t)/c_\tau$, it follows that if at least a $9/10$ fraction of slots in the interval have contention at most C_{low}, then $(L(t_1) - L(t)) \leq O(\mathcal{A} + \mathcal{J}) - \Omega(\tau)$.

Overview of Section 5.4. Combining the analyses of $N(t)$, $H(t)$, and $L(t)$, to analyze $\Phi(t)$. This section combines all three terms of the potential function to characterize the overall behavior of $\Phi(t)$. The key tools established in this section are Theorem 27 (stated

previously in Section 3.3) and Theorem 28, which allow us to argue that the potential will decrease (most of the time) and that, when this fails to occur, the amount by which it increases is bounded. Specifically, consider a size-τ interval with $\mathcal{A}$ packet arrivals and $\mathcal{J}$ jammed slots. Theorem 27 shows that $\Phi(t)$ decreases by $\Omega(\tau) - O(\mathcal{A}+\mathcal{J})$ w.h.p. in τ. Theorem 28 establishes tail bounds, proving that even when the high-probability bound of Theorem 27 fails, the probability that $\Phi(t)$ increases by more than $k\tau^2 + O(\mathcal{A}+\mathcal{J})$ is less than $\frac{1}{\text{poly}(\tau)} \cdot (1/2)^{\Theta(\log^2 k)}$.

Theorems 27 and 28 are the tools needed to fit our betting game, descussed next and in Section 5.5, and thereby argue that the potential is likely to decrease sufficiently across multiple intervals.

Theorem 28 (Tail bound on increase in $\Phi(t)$ over interval $\mathcal{I}$). *Consider an interval $\mathcal{I}$ of length $|\mathcal{I}| = \tau$ starting at time t and ending at time t'. Let $\mathcal{A}$ be the number of packet arrivals in $\mathcal{I}$. Then the probability that Φ increases by at least $\Theta(\mathcal{A}) + \Theta(k\tau^2)$ is at most $2^{-\Theta(c(\lg\tau\cdot\lg k+\lg^2 k))} \le (1/\tau^{\Theta(c)})\cdot 2^{-\Theta(\log^2 k)}$, where c is the constant parameter of the algorithm. That is,*

$$\Pr\left[(\Phi(t') - \Phi(t)) \ge \Theta(\mathcal{A}) + \Theta(k\tau^2)\right] \le \left(\frac{1}{\tau^{\Theta(1)}}\right) \cdot 2^{-\Theta(\log^2 k)}.$$

Overview of Section 5.5. Using $\Phi(t)$ to prove throughput via a betting-game argument. The analysis so far establishes progress guarantees over sufficiently large intervals in the form of Theorems 27 and 28. Here, we show how to apply these theorems to give upper bounds on the potential over the execution with high probability in the total number of packets and jammed slots.

Since the adversary is adaptive, we have to be careful in combining bounds across intervals. The adversary can use the results of earlier intervals in choosing new arrivals and jamming, which affects the size of later intervals. To reason about this process, we reframe it in a setting that resembles a random walk, which we describe below in a ***betting game***. Our analysis of this game then allows us to analyze the implicit throughput (recall Section 1.1).

The Betting Game. We first summarize the betting game and then later relate it to the backoff process. The adversary corresponds to a ***bettor*** who makes a series of bets. Each bet has a size equal to the duration τ. The bettor also has some amount of money, which is initially 0 dollars. When the bettor loses a bet, the bettor loses some money, and when the bettor wins, the bettor wins some money. (The amounts won or lost are specified below as a function of the size of the bet.) Additionally, at any time, the bettor may choose to receive a ***passive income***. The passive income is added to the bettor's wealth. The total amount of passive income taken, however, means that the bettor must play the game longer. The game begins when the bettor first takes some passive income, and the game does not end until either the bettor goes broke or the bettor has resolved bets totaling $\Omega(P)$ size, where P is the passive income received, whichever comes first. The bettor's goal is to complete the game without going broke. Importantly, although the bettor can always choose to take more passive income, doing so increases the total play time.

We set the details of the betting game to mirror the backoff process. Each bet corresponds to an interval. Passive income during a bet corresponds to the number of arrivals and jammed slots during the interval. Money corresponds to potential.

The bettor loses a size-τ bet with probability at least $1 - \frac{1}{\text{poly}(\tau)}$. If the better loses the size-τ bet, it loses $\Theta(\tau)$ dollars. This loss corresponds to the high-probability event (in τ) of Theorem 27. The bettor wins a size-τ bet with probability $O(1/\text{poly}(\tau))$. If the bettor wins the bet, it gets $\Theta(\tau^2)$ dollars, plus Y ***bonus dollars***, where Y is a random variable such that $\Pr[Y \ge k\tau^2] \le \frac{1}{\text{poly}(\tau)} \cdot 2^{-\Theta(\log^2 k)}$; these winnings correspond to tail bound of Theorem 28. (Of course, during each bet, the bettor can also gain passive income for arrivals and jammed slots.)

We pessimistically give the bettor the power to choose arbitrary bet sizes (subject to a minimum interval size, which itself is determined by $w_{\min}$), and the bettor is even allowed to place bets whose loss would cause the bettor to end with negative money. (In the actual backoff process, the interval sizes are dictated by the current state of the system, and not entirely under the control of the adversary.)

The rules of betting game are set pessimistically (in favor of the bettor) such that when the bettor wins, $\Phi(t)$ increases more slowly than the bettor's wealth increases, and when the bettor loses, $\Phi(t)$ decreases at least as fast as the bettor's wealth decreases. Therefore, this betting game stochastically dominates the potential function.

The takeaway is that at any point t, the bettor's wealth is an upper bound on $\Phi(t)$. Because $\Phi(t)$ is an upper bound on the number of packets in the system, the bettor going broke corresponds to all packets succeeding. We thus obtain good implicit throughput, because there must either be many jammed slots or packet arrivals, or there must be many packets succeeding, leading to inactive slots.

Upper bounding the bettor's maximum wealth/potential and showing $\Omega(1)$ implicit throughput. In Lemma 29, we provide a high-probability upper bound on the bettor's maximum wealth and the amount of time until it goes broke, which corresponds to there being no packets in the system.

Lemma 29 (The bettor loses the betting game). *Suppose the bettor receives P dollars of passive income. Then with high probability in P, the bettor never has more than $O(P)$ dollars across the execution. Moreover, the bettor goes broke within $O(P)$ active slots, with high probability in P.*

We briefly explain here how Lemma 29 implies implicit throughput. Consider a time horizon t, and suppose that the bettor has received $P = t/c$ dollars from passive income, for constant c matching the big-O of the lemma. Then from Lemma 29, with high probability in t/c, the bettor goes broke within $c \cdot t/c = t$ time; that is, there are no active packets at time t. We thus obtain the $\Omega(1)$ throughput result of Theorem 3.

Overview of Section 5.6. Channel access/energy bounds. In this section, we establish energy bounds. Two of the theorem statements, namely Theorems 34 and 38; the rest appear in Section 5.6. Theorems 34–38 are proved via properties of our potential function. (Several additional useful lemmas about the potential, not highlighted above, do appear in Section 5.5). So far, we have primarily motivated $\Phi(t)$ as a tool for proving throughput bounds, but $\Phi(t)$ also enables channel-access bounds.

Theorem 34 gives energy bounds in the finite case against an adaptive adversary. Specifically, if the stream has N packets and

$\mathcal{J}$ jammed slots, then w.h.p. each packet accesses the channel at most $\text{polylog}(N + \mathcal{J})$ times. The proof structure is as follows: Our upper bound on $\Phi(t)$ immediately gives an upper bound on a packet's maximum window size: $w_{\max}(t) = O(\text{poly}(\Phi(t)) = O(\text{poly}(N + \mathcal{J}))$. Thus, if a packet accesses the channel too many times, then many of these accesses must have been listening during silent slots, so that the packet window can get smaller. However, by the structure of Low-Sensing Backoff, whenever a packet first chooses to listen, there is at least a $1/\text{polylog}(N + \mathcal{J})$ probability that it also sends. Thus, after $\text{polylog}(N + \mathcal{J})$ channel accesses when all other packets are silent, with high probability that packet has been transmitted.

Theorem 34 (Energy bound for finite case against adaptive adversary). *Consider an input stream with N packets and $\mathcal{J}$ jammed slots. Assume that the adversary is adaptive but not reactive. Then w.h.p. in $N + \mathcal{J}$, every packet accesses the channel at most $O(\log^4(N + \mathcal{J}))$ times.*

The corresponding proof illustrates one subtle design choice of Low-Sensing Backoff, which leads to an easier energy analysis. Specifically, a given packet's sending and listening probabilities are correlated: if a packet sends, then it has already decided to listen (but, of course, a packet can listen without deciding to send). We conclude by observing that, with an adaptive adversary, all packets have good channel-access bounds.

Theorem 35 gives an analogous result for an adversary that is both adaptive and reactive. By the very nature of a reactive adversary, there is no possibility of good per-packet bounds on channel accesses. (For example, a reactive adversary could target a specific packet and reactively jam whenever it sees this packet try to transmit.) However, interestingly, the amortized channel-access bounds are still good. This is because the reactive adversary only learns about sending on the channel and can react instantaneously; it does not learn whether a packet is listening in the current slot. Thus, a targeted packet can still reduce its window (as the other packets do) and it will succeed in sending unless the adversary does significant jamming. For example, consider the special case where the targeted packet is the only packet remaining. Then, unless the adversary (which does not sense when a packet will listen) jams a large number of slots, this packet will correctly back on and transmit successfully.

Theorems 36 and 37 generalize Theorems 34 and 35 to the adversarial-queuing setting with granularity S and sufficiently small arrival rate λ. The main tool is Lemma 32, which allows us to transform the adversarial queuing case into finite instances that are not very large. Theorem 38 applies to infinite streams with arbitrary arrivals.

Theorem 38 (Channel access bounds for infinite case against adaptive and reactive adversaries). *Suppose that up until time t there have been N_t packet arrivals and $\mathcal{J}_t$ jammed slots.*

- *Consider an adaptive adversary that is not reactive. Then w.h.p. in $\mathcal{J}_t + N_t$, each packet makes $O(\log^4(\mathcal{J}_t + N_t))$ channel accesses before time t.*
- *Consider and adaptive adversary that is reactive. Then w.h.p. in $\mathcal{J}_t + N_t$, a particular packet accesses the channel at most $O((\mathcal{J}_t + 1)\log^3(N_t + \mathcal{J}_t) + \log^4(N_t + \mathcal{J}_t))$ times. Moreover, the average number of channel accesses is $O((\mathcal{J}_t/N_t + 1)\log^4(N_t + \mathcal{J}_t))$.*

4 CONCLUSION

We have provided a simple contention-resolution algorithm that achieves constant throughput with full energy efficiency (i.e., low sending *and* listening complexity), despite a jamming adversary. This resolves in the affirmative two open questions about whether full-energy efficiency is possible *at all* in the popular ternary-feedback model, and whether it remains possible despite jamming.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CCF-1918989, CCF-2106759, CCF-2144410, CCF-2247577, CCF-2106827, and by Singapore MOE-T2EP20122-0014.

REFERENCES

[1] Norman Abramson. 1970. The ALOHA System: Another Alternative for Computer Communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference* (Houston, Texas). Association for Computing Machinery, New York, NY, USA, 281–285.

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

[3] Hesham Al-Ammal, Leslie Ann Goldberg, and Phil MacKenzie. 2000. *Binary Exponential Backoff Is Stable for High Arrival Rates.* Springer Berlin Heidelberg, Berlin, Heidelberg, 169–180.

[4] Hesham Al-Ammal, Leslie Ann Goldberg, and Phil MacKenzie. 2001. An improved stability bound for binary exponential backoff. *Theory of Computing Systems* 34, 3 (2001), 229–244.

[5] David J. 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.

[6] 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.

[7] 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)*. ACM, Toronto, Canada, 45–54.

[8] Michael A. Bender, Martin Farach-Colton, Simai He, Bradley C. Kuszmaul, and Charles E. Leiserson. 2005. Adversarial Contention Resolution for Simple Channels. In *Proc. 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, Las Vegas, USA, 325–332.

[9] Michael A Bender, Jeremy T Fineman, Seth Gilbert, John Kuszmaul, and Maxwell Young. 2023. Fully Energy-Efficient Randomized Backoff: Slow Feedback Loops Yield Fast Contention Resolution. arXiv preprint arXiv:2302.07751. Full version of this paper: https://doi.org/10.48550/arXiv.2302.07751.

[10] Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Maxwell Young. 2019. Scaling Exponential Backoff: Constant Throughput, Polylogarithmic Channel-Access Attempts, and Robustness. *J. ACM* 66, 1 (January 2019), 6:1–6:33.

[11] Michael A. Bender, Tsvi Kopelowitz, William Kuszmaul, and Seth Pettie. 2020. Contention Resolution without Collision Detection. In *Proc. 52st Annual ACM Symposium on the Theory of Computing (STOC)*. ACM, Chicago, USA, 105–118.

[12] Michael A. Bender, Tsvi Kopelowitz, Seth Pettie, and Maxwell Young. 2016. Contention Resolution with Log-Logstar Channel Accesses. In *Proc. 48th Symposium on the Theory of Computing (STOC)*. ACM, Cambridge, USA, 499–508.

[13] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. 2001. Adversarial queuing theory. 48, 1 (2001), 13–38.

[14] Yi-Jun Chang, Wenyu Jin, and Seth Pettie. 2019. Simple Contention Resolution via Multiplicative Weight Updates. In *2nd Symposium on Simplicity in Algorithms (SOSA)*, Vol. 69. Schloss Dagstuhl, San Diego, USA, 16:1–16:16.

[15] Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. 2019. Exponential Separations in the Energy Complexity of Leader Election. *ACM Trans. Algorithms* 15, 4 (2019), 49:1–49:31.

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

[17] Tapiwa M Chiwewe, Colman F Mbuya, and Gerhard P Hancke. 2015. Using cognitive radio for interference-resistant industrial wireless sensor networks: An overview. *IEEE Transactions on Industrial Informatics* 11, 6 (2015), 1466–1481.

[18] Bogdan S. Chlebus, Gianluca De Marco, and Dariusz R. Kowalski. 2016. Scalable Wake-up of Multi-channel Single-hop Radio Networks. *Theoretical Computer Science* 615, C (Feb. 2016), 23–44.

[19] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2006. Adversarial queuing on the multiple-access channel. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, Denver, Colorado, USA, 92–101.

[20] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2007. Stability of the multiple-access channel under maximum broadcast loads. In *Proceedings of the Symposium on Self-Stabilizing Systems (SSS)*. Springer, Springer-Verlag, Paris, France, 124–138.

[21] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2009. Maximum throughput of multiple access channels in adversarial environments. *Distributed Computing* 22, 2 (2009), 93–116.

[22] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2012. Adversarial Queuing on the Multiple Access Channel. *ACM Transactions on Algorithms* 8, 1 (2012), 1–31.

[23] Federal Communications Commission. 2014. FCC 14-55: Notice of Apparent Liability for Forfeiture of Illegal Operation of Signal Jamming Device. https://apps.fcc.gov/edocs_public/attachmatch/FCC-14-55A1.pdf.

[24] Gianluca De Marco, Dariusz R Kowalski, and Grzegorz Stachowiak. 2022. Contention resolution without collision detection: constant throughput and logarithmic energy. In *Proceedings of the 36th International Symposium on Distributed Computing (DISC)*. Schloss Dagstuhl, Augusta, Georgia, USA, 1–21.

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

[26] Laura Marie Feeney and Martin Nilsson. 2001. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001*, Vol. 3. IEEE, Anchorage, AK, USA, 1548–1557.

[27] Daquan Feng, Chenzi Jiang, Gubong Lim, Leonard J Cimini, Gang Feng, and Geoffrey Ye Li. 2012. A survey of energy-efficient wireless communications. *IEEE Communications Surveys & Tutorials* 15, 1 (2012), 167–178.

[28] Jeremy T. Fineman, Calvin Newport, and Tonghe Wang. 2016. Contention Resolution on Multiple Channels with Collision Detection. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, Chicago, IL, USA, 175–184.

[29] Seth Gilbert, Valerie King, Seth Pettie, Ely Porat, Jared Saia, and Maxwell Young. 2014. (Near) Optimal Resource-competitive Broadcast with Jamming. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, Prague, Czech Republic, 257–266.

[30] Leslie Ann Goldberg. 2000. Notes on Contention Resolution. (2000). http://www.dcs.warwick.ac.uk/~leslie/contention.html

[31] 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.

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

[33] Albert G Greenberg and Schmuel Winograd. 1985. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *Journal of the ACM (JACM)* 32, 3 (1985), 589–596.

[34] Maurice Herlihy and J. Eliot B. Moss. 1993. Transactional Memory: Architectural Support for Lock-Free Data Structures. In *Proceedings of the 20th International Conference on Computer Architecture*. ACM, San Diego, CA, USA, 289–300.

[35] IEEE. 2016. IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks – Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. , 3534 pages.

[36] Yonggang Jiang and Chaodong Zheng. 2022. Robust and Optimal Contention Resolution without Collision Detection. In *Proceedings of the Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, Philadelphia, PA, USA, 107–118.

[37] Frank P. Kelly. 1985. Stochastic Models of Computer Communication Systems. *Journal of the Royal Statistical Society, Series B (Methodological)* 47, 3 (1985), 379–395.

[38] Frank P Kelly and Iain M MacPhee. 1987. The number of packets transmitted by collision detect random access schemes. *The Annals of Probability* 15, 4 (1987), 1557–1568.

[39] Mounib Khanafer, Mouhcine Guennoun, and Hussein T Mouftah. 2013. A survey of beacon-enabled IEEE 802.15. 4 MAC protocols in wireless sensor networks. *IEEE Communications Surveys & Tutorials* 16, 2 (2013), 856–876.

[40] Valerie King, Seth Pettie, Jared Saia, and Maxwell Young. 2018. A resource-competitive jamming defense. *Distributed Computing* 31, 6 (2018), 419–439.

[41] James F. Kurose and Keith Ross. 2002. *Computer Networking: A Top-Down Approach Featuring the Internet* (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[42] William Kuszmaul and Qi Qi. 2021. The Multiplicative Version of Azuma's Inequality, with an Application to Contention Analysis. arXiv preprint arXiv:2102.05077.

[43] Mads Lauridsen, Rasmus Krigslund, Marek Rohr, and Germán Madueno. 2018. An empirical NB-IoT power consumption model for battery lifetime estimation. In *IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, Porto, Portugal, 1–5.

[44] Yuan Li, Wei Ye, and John Heidemann. 2005. Energy and latency control in low duty cycle MAC protocols. In *IEEE Wireless Communications and Networking Conference, 2005*, Vol. 2. IEEE, New Orleans, LA, USA, 676–682.

[45] Junyu Liu, Min Sheng, Lei Liu, and Jiandong Li. 2017. Interference management in ultra-dense networks: Challenges and approaches. *IEEE Network* 31, 6 (2017), 70–77.

[46] Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. 2022. Time and Energy Efficient Contention Resolution in Asynchronous Shared Channels. arXiv:2209.14140

[47] Christophe J Merlin and Wendi B Heinzelman. 2010. Duty cycle control for low-power-listening MAC protocols. *IEEE Transactions on Mobile Computing* 9, 11 (2010), 1508–1521.

[48] Robert M. Metcalfe and David R. Boggs. 1976. Ethernet: Distributed Packet Switching for Local Computer Networks. *Commun. ACM* 19, 7 (July 1976), 395–404.

[49] Aristides Mpitziopoulos, Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. 2009. A Survey on Jamming Attacks and Countermeasures in WSNs. *IEEE Communications Surveys & Tutorials* 11, 4 (2009), 42–56.

[50] AP News. 2014. Marriott fined $600,000 for jamming guests' Wi-Fi. https://apnews.com/article/a63d6eaa5c1a4769bc786bb4fe456231.

[51] Pulse News. 2014. Angry priest installs phone jamming device in church to stop calls. https://www.pulse.ng/communities/religion/enough-is-enough-angry-priest-installs-phone-jamming-device-in-church-to-stop-calls/xq02ts1.

[52] Adrian Ogierman, Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2018. Sade: competitive MAC under adversarial SINR. *Distributed Computing* 31, 3 (01 Jun 2018), 241–254.

[53] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V Krishnamurthy. 2010. Denial of service attacks in wireless networks: The case of jammers. *IEEE Communications Surveys & Tutorials* 13, 2 (2010), 245–257.

[54] J. Polastre, R. Szewczyk, and D. Culler. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, Boise, ID, USA, 364–369.

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

[56] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2011. Competitive and Fair Medium Access Despite Reactive Jamming. In *Proceedings of the 31st International Conference on Distributed Computing Systems*. IEEE, Minneapolis, MN, USA, 507–516.

[57] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2012. Competitive and Fair Throughput for Co-Existing Networks Under Adversarial Interference. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, Madeira, Portugal, 291–300.

[58] Walter A. Rosenkrantz. 1984. Some Theorems on the Instability of the Exponential Back-Off Protocol. In *Proceedings of the Tenth International Symposium on Computer Performance Modelling, Measurement and Evaluation (Performance '84)*. North-Holland Publishing Co., NLD, 199–205.

[59] William N Scherer III and Michael L Scott. 2005. Advanced contention management for dynamic software transactional memory. In *Proceedings of the 24th ACM symposium on Principles of Distributed Computing (PODC)*. ACM, Las Vegas, NV, USA, 240–248.

[60] Nah-Oak Song, Byung-Jae Kwak, and Leonard E. Miller. 2003. On the Stability of Exponential Backoff. *Journal of Research of the National Institute of Standards and Technology* 108, 4 (2003), 289–297.

[61] Chicago Tribune Steve Schmadeke. 2016. Lawyer: Man accused of jamming calls on Red Line 'disturbed by people talking around him'. http://www.chicagotribune.com/news/local/breaking/ct-cell-phone-jamming-red-line-20160309-story.html.

[62] Adam Wierman and Takayuki Osogami. 2003. A unified framework for modeling TCP-Vegas, TCP-SACK, and TCP-Reno. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*. IEEE, Orlando, FL, USA, 269–278.

[63] Matthias Wilhelm, Ivan Martinovic, Jens B Schmitt, and Vincent Lenders. 2011. Short paper: Reactive jamming in wireless networks: How realistic is the threat?. In *Proceedings of the fourth ACM conference on Wireless network security*. ACM, Hamburg Germany, 47–52.

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

[65] Yang Xiao. 2005. Performance Analysis of Priority Schemes for IEEE 802.11 and IEEE 802.11e Wireless LANs. *Wireless Communications, IEEE Transactions on* 4, 4 (July 2005), 1506–1515.

[66] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. 2005. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *MobiHoc*. ACM, Urbana-Champaign, IL, USA, 46–57.

[67] Wenjie Yang, Mao Wang, Jingjing Zhang, Jun Zou, Min Hua, Tingting Xia, and Xiaohu You. 2017. Narrowband wireless access for low-power massive internet of things: A bandwidth perspective. *IEEE wireless communications* 24, 3 (2017), 138–145.

[68] Wei Ye, Fabio Silva, and John Heidemann. 2006. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, Boulder, CO, USA, 321–334.