Contention Resolution for Coded Radio Networks

Michael A. Bender
Stony Brook University
Stony Brook, New York, USA
bender@cs.stonybrook.edu

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

Fabian Kuhn
University of Freiburg
Freiburg, Germany
kuhn@cs.uni-freiburg.de

John Kuszmaul
Yale University
New Haven, Connecticut, USA
john.kuszmaul@yale.edu

Muriel Médard
Massachusetts Institute of
Technology
Cambridge, Massachusetts, USA
medard@mit.edu

ABSTRACT

Randomized backoff protocols, such as exponential backoff, are a powerful tool for managing access to a shared resource, often a wireless communication channel (e.g., [1]). For a wireless device to transmit successfully, it uses a backoff protocol to ensure exclusive access to the channel. Modern radios, however, do not need exclusive access to the channel to communicate; in particular, they have the ability to receive useful information even when more than one device transmits at the same time. These capabilities have now been exploited for many years by systems that rely on interference cancellation, physical layer network coding and analog network coding to improve efficiency. For example, Zigzag decoding [56] demonstrated how a base station can decode messages sent by multiple devices simultaneously.

In this paper, we address the following question: *Can we design a backoff protocol that is better than exponential backoff when exclusive channel access is not required*. We define the Coded Radio Network Model, which generalizes traditional radio network models (e.g., [30]). We then introduce the Decodable Backoff Algorithm, a randomized backoff protocol that achieves an optimal throughput of $1 - o(1)$. (Throughput 1 is optimal, as simultaneous reception does not increase the channel capacity.) The algorithm breaks the constant throughput lower bound for traditional radio networks [47–49], showing the power of these new hardware capabilities.

CCS CONCEPTS

• **Theory of computation** → *Online algorithms*; **Design and analysis of algorithms**; *Distributed algorithms*; • **Networks** → **Network algorithms**.

KEYWORDS

contention resolution; randomized backoff; coded networks; scheduling

1 INTRODUCTION

Randomized **backoff** protocols [1–3, 47, 63, 79], such as exponential backoff [79], are a powerful tool for managing access to a shared resource, often a communication channel. These protocols are used prominently in wireless networks from AlohaNet [2, 3, 25, 91] to 802.11 [1] and beyond [26]. Wireless devices typically need to transmit their messages to a base station, and in order for

a device's transmission to be successful, the device must have **exclusive access** to the channel, that is, no other wireless device can be transmitting. If more than one device transmits simultaneously, there is a **message collision** and the base station receives indecipherable noise [47, 53, 65, 74, 79, 98].

The problem of designing backoff protocols is traditionally formalized as the **contention-resolution** problem. The devices correspond to agents that arrive over time, each with a message that needs to be transmitted. To perform the transmission, the device requires exclusive access to the channel [2, 3, 30, 47, 53, 65, 74, 79, 98]. Time is subdivided into synchronized **slots**, which are sized to fit a message, and the objective is to maximize the **utilization** or **throughput** of the channel, which is roughly (but not exactly) the fraction of slots that successfully transmit messages. For example, traditional binary exponential backoff yields $\Theta(1/\log n)$ throughput with adversarial packet injection [16]; for stochastic packet injections, there is much work analyzing the arrival rates under which binary exponential backoff is stable or unstable [7, 57, 62].[1] Newer backoff protocols can achieve $\Theta(1)$ throughput [15, 19–23, 27] even under adversarial packet injection.

Modern devices do not need exclusive access to the channel to broadcast. An exciting development over the last decade has been improvements in modern radio hardware, in particular, the ability to receive useful information when more than one device transmits at the same time [11, 26, 38, 56, 61, 69–71, 94]. This information is not sufficient to immediately recover the messages sent; however, when a base station receives enough information about a set of messages, it can then decode the original messages. The point is that on modern hardware, backoff protocols do not need to guarantee exclusive access to the channel, because exclusive access is no longer necessary.

In fact, these capabilities have now been exploited for many years by systems that rely on interference cancellation [11, 26, 61, 94], physical layer network coding [38, 41, 70, 71], and analog network coding [69] to improve efficiency. For example, the celebrated paper on Zigzag decoding [56] showed how a base station can decode messages sent by multiple devices simultaneously. (In one scenario cited in their paper, this reduced packet loss from approximately 72% to less than 1%!)

---

[1]In fact, there is much work on stochastic arrival models [4, 6, 47, 50, 52, 57, 57, 65, 65, 72, 79, 80, 83, 84, 92].

ZigZag decoding was built on top of 802.11, i.e., exponential backoff; it showed that, in practice, improved signal-decoding techniques can yield significant improvements, even while using traditional exponential backoff for contention resolution.

Coding-compatible contention resolution. In this paper, we address the following question, taking an algorithmic approach: Can we design a backoff protocol that is better than exponential backoff for this setting where exclusive channel access is not required, that is, when devices can have simultaneous transmissions? To do so, we define the ***Coded Radio Network Model*** (see Section 2), which generalizes traditional radio network models [30] (where devices *do* need exclusive access to the channel to transmit). We then ask: what is the best throughput that can be achieved?

To see an example of the throughput achieved by some standard approaches, consider the case where the base station can only decode a message if there is exactly one transmitter. Then, if there are n devices and each device broadcasts in each round with probability $1/n$, the system achieves throughput $1/e$ (and many consider this to be the ideal achievable throughput in many situations). With binary exponential backoff, if n devices begin the protocol at the same time, it takes $\Theta(n \log n)$ time for all the packets to complete, and the throughput achieved is $1/\log n$ [16]. Recent work has shown how to achieve $\Theta(1)$ throughput even in an adversarial setting [20, 27, 28], and Chang, Jin, and Pettie [27] showed how to achieve throughput $1/e - O(\varepsilon)$, for any $\varepsilon > 0$.

At the other end of the spectrum, several papers have proved lower bounds that establish the best achievable throughput in different situations. For example, in a "full sensing" model, no protocol can achieve better than 0.568 throughput [47, 96]; in an "acknowledgment-based" model, no protocol can achieve better than 0.530045 throughput [47–49].

This raises the question: *what is the best achievable throughput in a system that uses modern techniques to decode simultaneous signals?* If we can send an arbitrary number of messages simultaneously, it may seem as if there is no inherent limit. This is not the case. There are two main reasons, one pragmatic and one information theoretical. First, from a practical perspective, there is a limit to the number of simultaneous signals that a base station can usefully interpret. For each signal received, there is noise, fading, frequency offset, inter-symbol interference, etc., all of which means that the more simultaneous signals, the less useful information the base station receives (see, e.g., [95], Chapter 3). Thus, there is some ***decoding threshold*** κ where if there are more than κ simultaneous transmissions, the base station learns nothing useful. A key implication here is that there is a maximum achievable throughput, and contention resolution is still important—even though we no longer need to guarantee exclusive channel access for the transmitter.

From the perspective of information theory, there is an even stronger limitation: none of the techniques we are considering (e.g., iterative interference cancellation) has increased the actual capacity of the communication channel.[2] It simply provides a method for using that capacity more efficiently. For example, if two messages are being transmitted simultaneously, it will then take twice as long to receive and decode them. This is observed in practice in ZigZag Decoding [56], where the authors state: "when senders collide, ZigZag attains the same throughput as if the colliding packets were a priori scheduled in separate time slots." The conclusion, then, is that the maximum achievable throughput is still 1.

Results. In this paper, we develop a randomized backoff protocol designed for a system that can support multiple simultaneous transmissions. Our model captures the fact that the base station can only receive useful information if there are $\leq \kappa$ simultaneous transmissions, and it reflects the fact that (conceptually) if ℓ messages are transmitted simultaneously, the base station only receives roughly $1/\ell$ of each original message. The new algorithm achieves optimal throughput arbitrarily close to 1, approaching 1 as κ grows. More precisely, it achieves a throughput of $1 - \Theta(1/\log \kappa)$. (And in doing so, it shows that the increased power of simultaneous transmission is non-trivial, as it breaks the constant-throughput lower bounds [47–49, 96].)

The basic idea of the protocol is relatively simple: it uses a backon-backoff mechanism to construct groups of messages that will broadcast simultaneously until they are decoded. The backoff-backon mechanism is tuned to yield groups of size at most (the decoding threshold) κ, and a simple "admission control" mechanism prevents newly arrived packets from disrupting the ongoing protocol. (Ensuring that the same packets broadcast together repeatedly is also critical to protocols like ZigZag Decoding [56]—it is the repeated overlaps that make simultaneous decoding easier.) Our analysis relies on a potential-function argument, which shows that, at any given time, we are either decoding groups of packets or moving closer to a good group structure.

The key challenge is getting throughput $1 - o(1)$. Even wasting a few rounds to perform backoff may render our goals impossible to achieve. For example, if there are x packets injected into the system, and we spend some $\Theta(x)$ rounds running backoff to construct batches of size κ, then our throughput will not be close to 1. Indeed, a classical backoff protocol like exponential backoff [79] spends at least $\Theta(x)$ time even just growing its window to size $\Theta(x)$. Thus, in both the protocol design and the analysis, we have to be particularly careful to prevent constant factors from creeping in!

To show good throughput, we focus on the *rate* at which packets can be added to the system without creating a large backlog of undelivered packets. If packets arrive at a rate equal to the channel capacity, and yet there is never a backlog, then we conclude that the backoff protocol is delivering a throughput equal to the channel capacity. (We are not interested in executions where there are very few packets—the throughput may look low simply because there is nothing to do.) We allow for arbitrary adversarial packet arrival, i.e., packet arrivals can be bursty, smooth, or whatever. (There is no assumption of random packet arrivals.)

To show that we achieve throughput arbitrarily close to 1, we assume that for some window size w, there are at most $(1 - o(1))$ packet arrivals in every window of size at least w, i.e., at some level of granularity the arrivals do not exceed the channel capacity. We show that in this case, there is never a large backlog of packets

---

[2]Typically, the capacity of a wireless channel depends on other factors, e.g., the transmission frequency, the modulation scheme, etc. As we do not want to address those issues in this paper, we simply define one time slot as the time it takes to send a single message using the system at hand.

in the system, i.e., the system continues to deliver packets. In addition, we show that there is no starvation, i.e., that packets do not stagnate in the system too long and will eventually be delivered.

THEOREM 1. *Suppose that in every window of size w, for some $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln \kappa}\right)w$ packet arrivals. Then the Decodable Backoff Algorithm guarantees the following two conditions in the Coded Radio Network Model:*
(i) At any given time, the number of packets in the system is at most $2w$, with probability $\geq 1 - 1/\mathrm{poly}(w)$.
(ii) A given packet is delivered within $O(\sqrt{\kappa}w \ln^3 w)$ time slots, with probability $\geq 1 - 1/\mathrm{poly}(w)$.

These results are proved in Theorem 11 and Theorem 15.

One thing to note is that the window size w is not known to the algorithm—it is simply an analytic tool. Thus the execution can have entirely arbitrary packet arrivals and the theorem will hold if there exists some w for which the arrival rate is satisfied. Another note is that, despite the high probability bounds, the execution is not limited to polynomial lengths; the claims will hold for any time t, even in a very long execution. Finally, observe that for packet latency, the best possible result is $O(w)$: if the adversary injects w packets in round 1, it will necessarily take at least w rounds for all of them to complete. (In fact, we consider the batch case in Theorem 16.)

Other related work. There is a long history of algorithmic research analyzing the performance of backoff protocols, looking at different aspects. As in this paper, the primary goal has been to analyze packet delivery rates, whether in the form of throughput, total time, or stability [16, 17, 24, 27, 37, 52, 54, 58, 64, 64, 76, 79, 85, 93]. An alternative metric has been looking at how long it takes to deliver *one* message [31, 32, 35, 39, 40, 42, 45, 66, 66–68, 78, 97].

Backoff has been studied in a wide variety of different models. Many of the above papers have assumed that radios can detect collisions; some work has focused on the case where there is no collision detection [21, 28, 36, 37]. The large majority of these protocols have been randomized, but there has been some consideration of deterministic protocols [37, 73, 75, 77]. A few papers have focused on more realistic wireless models that capture signal-to-noise ratio [43, 44]. There has been some significant recent work on designing backoff protocols that are robust to noise and/or jamming [8, 9, 12, 15, 20, 31, 81, 82, 86–90]. A critical modelling question is how packets are injected, e.g., in batches [13, 16, 18, 46, 51, 59, 60, 97], stochastically [4, 6, 50, 52, 57, 57, 65, 65, 72, 79, 80, 83, 84, 92], or via an adversarial queuing theory model [10, 13, 16, 18, 33, 34, 51, 55, 59, 97]. There has also been a variety of work looking at variations, e.g., the packets may have deadlines [5], or the packets may have different sizes [18].

Our algorithm is built on the multiplicative-weight [14] approach, which has long been used for contention resolution, e.g., in TCP congestion control [29]. More recently, Awerbuch, Richa, and Scheideler [15] used a multiplicative-weight approach to solve backoff in the presence of jamming and noise. And Chang, Jin, and Pettie [27] used multiplicative-weights updates to achieve $1/e - o(1)$ throughput.

2 CODED RADIO NETWORK MODEL

The goal of this section is to develop the Coded Radio Network Model. We begin with background on how radios transmit and receive signals, before continuing to present the model itself.

Physical-layer radios

Here we briefly describe how physical-layer radios work. (See, e.g., [95], for more details.) The packet to be sent is broken up into smaller pieces. Each of these pieces is then "modulated," i.e., mapped to a complex number. The mapping depends on the modulation scheme, which impacts the transmission rate. For example, a simple binary encoding scheme might map a 1 to $e^{0i} = (1, 0)$ and 0 to $e^{\pi i} = (-1, 0)$. These complex numbers correspond to waveforms which the radio transmits. (For example, in "phase-shift keying," the complex number determines the *phase* of the wave-form; for the binary encoding above, the phase is 0 to transmit a 1 and π to transmit a 0.)

The receiver can observe the waveforms, which have been partially corrupted during the transmission: the signals are attenuated (i.e., the amplitude reduced), the phase has shifted based on the distance between the transmitter and receiver, there is additional noise, etc. Despite these challenges, modern receivers are designed to synchronize with the transmitter, estimate the various error parameters, and transform the waveform back into a sufficiently close approximation of the original complex number. These are then demodulated, and the packet reconstructed.

A notable aspect is that the process is *additive*: if two complex number are transmitted simultaneously, the receiver is able to decode the *sum* of the original complex numbers (each partially corrupted, as before, and with noise added). This sum can provide useful information about the original messages; for example, if you already know one of the messages, you can subtract it from the sum and recover the second message. This is famously exploited in systems that rely on interference cancellation, e.g., [11, 61, 70, 71].

Too many simultaneous transmissions, however, will make this demodulation increasingly difficult because of the noise and signal corruption that is added while the messages are in flight.

Our model

In this section, we describe the Coded Radio Network Model, which models radios that can receive multiple packets transmitted simultaneously, while respecting the information-theoretic capacity of the channel and the decoding limits inherent to modern hardware.

We assume that time is divided into (synchronized) slots, each of which is sufficiently large to send a single packet. At any given time, there are a collection of packets to be sent in the system. (Our discussion is going to focus on the *packets*, not the devices sending them, which may lead to some strange elocutions in which a message may seem to have agency in transmitting itself.)

There is a base station that is intended to receive all the packets. In a more traditional contention-resolution model, the base station would receive a packet that was broadcast by itself (i.e., the packet broadcast itself) in a time slot. Here, when there are simultaneous transmissions, we will allow the base station to recover some

information (i.e., at a lower level in the stack, it recovers sum information).

We define κ as the ***decoding threshold*** for the system, i.e., the threshold determined by the hardware for how many simultaneous transmissions can yield useful information. We say that a time slot is ***good*** if there is at least one and at most κ transmissions in that time slot. We say that a time slot is ***bad*** if there are more than κ transmissions in that slot. If no packets broadcast in the slot, it is a ***silent*** slot. (Silent slots are neither good nor bad.) The packets in the system cannot tell which steps are good and bad. They can hear which slots are silent, however.

The execution will consist of ***decoding windows***, i.e., periods of time during which the base station has collected enough information to decode packets. Packets that have been decoded are delivered and leave the system. Since a base station only learns useful information during a good time slot, it can only decode packets sent during good time slots—and ignores packets sent only during bad time slots. We focus on decoding windows where *every* packet that is broadcast during a good time slot must be decoded.

A decoding window needs to be long enough that the base station can decode all the packets sent during good slots. The information-theoretic constraint implies that if we are to decode j packets, we are going to need at least j good time slots. (In the next section, we explain the linear-algebraic intuition why j time slots is sufficient.)

At the end of a successful decoding window, a ***decoding event*** occurs:

DEFINITION 1. *There is a **decoding event** of size j at time t if there exists a time window that begins with a good slot and ends at time t, such that:*

 (1) The window contains no decoding events.
 (2) Exactly j packets broadcast in the window in good slots.
 (3) The number of good slots in the window is at least j.

*That window is the **decoding window**.*

 (4) Decoding events occur iteratively over time, meaning a decoding event occurs the first time that these conditions are satisfied after the previous decoding event.

When a decoding event of size j occurs, the j packets that broadcast in good slots during the decoding window are received by the base station.

Every device knows when a successful decoding event occurs.

Examples. If three packets broadcast simultaneously for three rounds (and $3 \leq \kappa$), then those three rounds yield a decoding event of size 3. Alternatively, if all three packets (a, b, c) broadcast in time slot 1 simultaneously, while packets b and c broadcast in time slot 2 and packet c broadcasts alone in time slot 3, then after these three time slots a decoding event occurs and all three packets are delivered. Notice that this definition satisfies our intuition that it should always take at least j time slots to successfully deliver j packets.

This formalization of a decoding event does not take advantage of all of the information that the decoding technology may be able to obtain. For example, consider the schedule in which packets a and b broadcast together in time steps 1 and 3. In step 2 a single packet c broadcasts, i.e., there is a decoding event in the window

comprised of only step 2. According to Definition 1, we cannot have a decoding event for steps 1-3, because there was already a decoding event at step 2. Since decoding windows are required to be disjoint, the broadcasts from step 1 cannot be used for any future decoding window. Even though the broadcasts in steps 1 and 3 provide sufficient information to decode packets A and B, that information is lost.

In fact, this paper will only need (and have) decoding intervals of size $O(\kappa)$. In practice, decoding windows will also be limited in length, e.g., the base station should not have to remember and try to decode arbitrarily long sequences.

Practicalities

Here, we are going to give some intuition for how random linear network coding might be used to implement the model described above. (Complete physical-layer implementations are beyond the scope of this paper.)

From the perspective of linear network coding, each time slot is a column vector t with one entry for each possible transmitter: $t[j] = 1$ if device j broadcasts and $t[j] = 0$ if device j is silent. Over the collection of good time slots that form a decoding window, the collection of vectors forms a binary matrix T containing one column for each good slot. (We drop the empty and bad time slots.)

If device j is transmitting message $m[j]$, then due to the additive nature of radio transmission, the base station receives the (row) vector mT during the decoding window. If the matrix T is invertible, then the base station can now decode the packets sent during this decoding window by computing mTT^{-1}.

There are many techniques for ensuring that the matrices are invertible. For example, random linear coding multiplies each message by a random coefficient before it is sent. The resulting transmission matrix T is now a random matrix and hence likely to be invertible. An alternative is to randomize the slots in which packets broadcast to ensure that each column vector is unique (which is all that is needed for linear independence).

A key technical issue is how the base station learns the transmission matrix. That is, the base station needs to know which packets are being transmitted in which slots in order to decode the packets. And in the case of random linear network coding, the base station also needs to know the coefficients. Thus, there is a small amount of "control information" that the base station needs to know before the packets can be decoded.

This problem is easier to solve in the context of this paper, where we ensure that the same packets always broadcast together during each slot of a decoding window. (For example, consider the situation where packets (a, b, c) are broadcast together three times until a decoding event occurs.) This decision makes for both a simpler algorithm, but also one that is more compatible with existing physical layer systems. Indeed, the requisite control information can even be sent using the ZigZag Decoding techniques [56]. In the general case, other techniques are needed to communicate this control information, including error-correcting codes, lower communication rates, leveraging random offsets (as in ZigZag Decoding), etc.

While our discussion here has been brief, developing systems with these types of network coding capabilities has been and remains an active and exciting area of research; see e.g., [26, 38, 41, 69–71].

3 DECODABLE BACKOFF ALGORITHM

In this section, we describe the Decodable Backoff Algorithm, a protocol for achieving efficient channel utilization in the Coded Radio Network model.

Admission control. Our algorithm separates packets into two types: *active* and *inactive*. Newly joined packets start as inactive. Inactive packets listen but do not broadcast; active packets do both.

Silent steps trigger *packet activations*. When an inactive packet hears a silent slot, it becomes active. An active packet stays active until it successfully broadcasts and leaves the system; it never returns to being inactive.

Types of epochs. The algorithm divides time into *epochs*, where the same set of packets broadcast in each slot of the epoch. The length of an epoch can vary from 1 to κ slots (as specified below).

Each packet j has an associated *joining probability p_j*, which means that when a new epoch begins, packet j *joins the epoch* with probability p_j and then broadcasts (deterministically) in every slot of the epoch. A packet has positive joining probability if and only if it is active. (An inactive packet j has joining probability $p_j = 0$.) If a packet does not join an epoch, then it remains silent until the epoch is complete.

There are three triggers that causes an epoch to end, leading to three types of epochs:

(1) **A silent slot**—This results in a *silent epoch*. A silent slot implies that no packets joined the epoch.
(2) **A decoding event**—This results in a *successful epoch* where the packets that joined the epoch are decoded.
(3) **κ time slots have elapsed without a decoding event**—This results in a *overfull epoch*, as it implies that too many packets joined the epoch and these slots will not be part of a decoding window.

Epoch-joining probabilities. When a packet j first becomes active, its initial joining probability is

$$p_j = 1/\sqrt{\kappa}. \tag{1}$$

When an epoch ends, packet j updates p_j as follows.

$$p_j \leftarrow \begin{cases} p_j \cdot \kappa^{1/4} & \text{upon a silent epoch,} \\[2mm] \dfrac{p_j}{\kappa^{1/4}} & \text{upon an overfull epoch,} \\[2mm] p_j & \text{upon a successful epoch.} \end{cases} \tag{2}$$

Algorithm highlights.

The algorithm itself is concise, but delicate: at the end of an epoch, packets multiplicatively update their broadcast probabilities by a factor of $\kappa^{1/4}$. Packets join with broadcast probability $p_0 = 1/\sqrt{\kappa}$. Inactive packets only become active upon hearing a silent epoch. Our target contention is $c_* = \sqrt{\kappa}$. If contention is in the window $[\kappa^{1/4}, \kappa^{3/4}]$, we say that *contention is good*.

There are three important aspects that are critical to its success. First, the epoch structure enables coordination among the packets so that packets retransmit together, which enables them to be decoded.[3] (Contrast this structure with a protocol in which devices broadcast independently in every round; it is much less likely to get enough information on a collection of packets that can be simultaneously decoded.) Second, careful admission control is important because it ensures that newly arriving packets do not interfere with packets that are currently participating in epochs.

Third, and most critically, setting the p_j's is delicate, given the objective of $1 - o(1)$ throughput. It is critical that newly active packets start with a $o(1)$ joining probability; see Equation (1), as otherwise too much time might be spent backing off. (Recall that even a small number of overfull epochs is too many for $1 - o(1)$ throughput.) Moreover, it is critical that the p_js change rapidly. To put Equation (2) in context, the "contention" on the channel (expected number of broadcasts in a slot) changes much faster (exponentially) than with exponential backoff [79]. Contention even changes a $\omega(1)$-factor faster than the multiplicative-weight-update algorithm of Chang et al. [27].

4 POTENTIAL FUNCTION

Throughout the execution, we maintain a potential function $\Phi(t)$ that captures the state of the system at t and measures the progress toward delivering all packets. We will see that packet arrivals increase $\Phi(t)$ by $1 + o(1)$ per newly arrived packet (Lemma 5), that packets exiting the system successfully decrease $\Phi(t)$ by 1 per packet (Lemma 7), and that an epoch of length ℓ is either a successful epoch or causes contention to move closer to the desired contention of c_* with high probability, and will decrease $\Phi(t)$ by $\ell(1 - o(1))$ in those cases (Lemma 3, Lemma 9).

Recall that p_i is the joining probability for each packet i. We define the *contention c_t* $= \sum_i p_i$. We define $p_{\min}(t) = \min_{\{i : i \text{ is an active packet}\}} p_i$ to be the minimum probability of any *active* packet currently in the system. If there are no active packets in the system, we define $p_{\min}(t) = 1$. We define the *target contention c_** $= \sqrt{\kappa}$. N_t is the total number of packets in the system at time t, including both active and inactive packets. M_t counts the number of inactive packets in the system at time t.

The potential function consists of four components (named below).

$$\Phi(t) = N_t + \max\left\{0, \, 4\kappa \log_\kappa\left(\frac{c_t}{c_*}\right)\right\} + 4\log_\kappa\left(\frac{1}{p_{\min}(t)}\right) + \frac{5\,M_t}{\ln \kappa}.$$

(1) $N(t) = N_t$. Since no term can go negative, this term implies that if the potential is small, there cannot be many packets remaining in the system.
(2) $\mathbf{logC}(t) = \max\left\{0, \, 4\kappa \log_\kappa\left(\frac{c_t}{c_*}\right)\right\}$. When positive, the term indicates how far the contention is from ideal, i.e., how many epochs of length κ are needed to reach c_*.
(3) $s(t) = 4\log_\kappa\left(\frac{1}{p_{\min}(t)}\right)$. The term indicates how many silent epochs (of length 1) would be necessary to bring the packet with minimal join-probability up to join probability 1.

---

[3]It is also the case that simultaneous retransmissions simplify the lower physical-layer implementation, as discussed in Section 2. For example, ZigZag Decoding [56] similarly relies on simultaneous retransmissions for this reason.

(4) $u(t) = \frac{5M_t}{\ln \kappa}$. The term compensates for packet activations. Potential is stored in this function for each arrived, inactive packet. That potential then compensates for the increase in the second term caused by the rise in contention when the packets activate.

It is immediate that the potential function is always positive when there are packets in the system. Since all the terms are non-negative, it also is immediately true that $N(t) \leq \Phi(t)$, i.e., the potential function is an upper bound on the number of packets in the system. If we can show that the potential in the system remains low, then we can conclude that the number of packets in the system also remains low.

It is constructive to compare our potential function with the function from Chang et al. [27]. Similarly, to Chang et al., $\Phi(t)$ has terms that measure the number of packets, how far contention is from the algorithm's target contention, and the smallest broadcast probability. Our design of $\Phi(t)$ also takes into account that (1) epochs can be different lengths, (2) there are transitions from inactive to active packets, (3) our multiplicative-weight update rule is especially aggressive, and (4) only a $o(1)$ fraction of slots can be bad.

5 THROUGHPUT ANALYSIS

In this section, we will show that the Decodable Backoff Algorithm guarantees $1 - o(1)$ throughput. Most of the section is devoted to analyzing the potential function, specifically showing that it decreases by $1 - o(1)$ in expectation in every round where it is $> 6\kappa$, and increases by $1 + o(1)$ for every packet that is injected by the adversary). Thus, in Theorem 11, we can conclude that as long as no more than $w(1 - 5/\ln \kappa)$ packets arrive in a window of length w, then (with high probability) there are never more than $2w$ packets in the system.

We first observe that successful epochs correspond to decoding windows:

LEMMA 2. *An epoch is successful if and only if $\leq \kappa$ packets join the epoch. In that case, there is a decoding event at the end of the epoch and the decoding window corresponds exactly with the epoch.*

PROOF. If $\leq \kappa$ packets join the epoch, then those packets will broadcast together in each of the following $\leq \kappa$ rounds of the epoch. These broadcasts cause a decoding event at the end of the epoch which includes those packets.

The last slot in a decoding window must be a good slot. (If the last slot were bad, then the conditions would have been satisfied earlier and the decoding event would have occurred earlier.) If $> \kappa$ packets join an epoch, then every round in the epoch is bad, and hence no decoding event occurs at the end of the epoch. If no packets join an epoch, then again the last slot of the epoch is not good and so again, no decoding event occurs.

The decoding window cannot be a suffix of the successful epoch that it concludes for the following reason: every slot in a successful epoch is good (because $\leq \kappa$ packets joined the epoch); if the decoding event delivers j packets, then it must occur after exactly j slots of the epoch—no earlier, and no later. If there were even one slot in the epoch that preceded the decoding window, then, the entire decoding window could have moved one slot earlier.

The decoding window cannot extend to the left of the successful epoch that ends with the decoding event. That is because between the last decoding event and the beginning of this epoch, every slot is either bad or empty, as each of the intervening epochs is either overfull or silent. And, as per the definition of the decoding window, it must begin with a good slot.

Thus we conclude that the decoding window corresponds exactly with a successful epoch which $\leq \kappa$ packets joined. □

5.1 Analyzing the probability of an error epoch

DEFINITION 2. *We define an **error epoch** to be an epoch that is either a silent epoch that occurs when contention $c_t \geq \kappa^{1/4}$ or an overfull epoch that occurs when contention $c_t \leq \kappa^{3/4}$. Note that this is well-defined since contention is constant throughout an epoch.*

LEMMA 3. *An epoch beginning at time t is an error epoch with probability at most $1/2^{\Theta(\kappa^{1/4})}$.*

PROOF. An error epoch is a silent epoch that occurs when $c_t \geq \kappa^{1/4}$ or an overfull epoch that occurs when $c_t \leq \kappa^{3/4}$.

Applying Chernoff bounds, the probability of an epoch being silent (i.e. no packets joining the epoch) when $c_t > \kappa^{1/4}$ is at most $1/2^{\Theta(\kappa^{1/4})}$. Similarly, the probability that an epoch is overfull (i.e. more than κ packets join) when $c_t < \kappa^{3/4}$ is at most $1/\kappa^{1/4(\Theta(\kappa))} = 1/\kappa^{\Theta(\kappa)}$. Applying a union bound, the probability of an error epoch occurring is at most $1/2^{\Theta(\kappa^{1/4})}$. □

We now bound the number of error epochs during an interval.

LEMMA 4. *Let t_2 be a time slot. Then for all intervals $I = (t_1, t_2]$ where $t_1 < t_2$, of size $w = t_t - t_1$ satisfying $\sqrt{w} > \Theta(\kappa^{1/4})$, the number of error epochs occurring in I is at most $\sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$, for any fixed constant $c \geq 6$, with probability $\geq 1 - 1/\text{poly}(w)$.*

PROOF. Choose an arbitrary slot $t_1 < t_2$, and define interval $I = (t_1, t_2]$. Let j be the random variable denoting the number of error epochs that occur in I. By Lemma 3, $\mathbb{E}[j] \leq (t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$, as I contains at most $t_2 - t_1$ distinct epochs. Let X_i be an indicator random variable that is 1 iff the ith epoch in I is an error epoch. Note that $\mathbb{P}[X_i = 1] \leq 1/2^{\Theta(\kappa^{1/4})}$, even if we condition on the values of X_j for all $j \neq i$. Thus, we can apply Chernoff Bounds to find:

$$\mathbb{P}\left[j \geq \sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}\right] \leq 1/2^{\sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}}.$$

Union bounding over all t_1 from $t_1 = 0$ to $t_1 = t_2$, we find that the probability that the number of error epochs in I exceeds $\sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$ for any interval $I = (t_1, t_2]$, over all t_1, is at most

$$\sum_{t_1=0}^{t_2} 1/2^{\sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}} \leq (2/c)2^{\Theta(\kappa^{1/4}) - \sqrt{w}}.$$

The inequality follows due to the sum being over a geometric series. □

5.2 Evolution of the Potential Function

Consider an epoch of length ℓ during which i packets are inserted by the adversary, for any arbitrary (ℓ, i). We prove in this subsection that error epochs are the only epochs in which the potential $\Phi(t)$ fails to decrease by at least $\ell(1 - 1/\kappa) - i(1 + 5/\ln \kappa)$ (Lemma 9).

First, we analyze the impact of packets joining the system on the potential function. This takes place in two steps. First, they arrive and are inactive. This only increases the first and last terms of the potential function:

LEMMA 5. *Each packet that arrives during in slot t increases the potential by $\Phi(t)$ by $1 + 5/\ln \kappa$.*

PROOF. Newly inserted packets arrive as inactive packets, and thus they have no impact on contention or $p_{\min}$. The only terms they affect are the first and last, which increase by 1 and $5/\ln \kappa$, respectively, per newly inserted packet. □

Next, we consider what happens to the potential function when packets activate. This activation occurs only during a silent epoch, and can impact only the second, third, and fourth terms of the potential function:

LEMMA 6. *If $\Phi(t) > 6\kappa$ and slot t is part of a non-error epoch, then packet activations do not increase $\Phi(t)$. More precisely, packet activations in slot t increase $\Phi(t)$ only in the following situations:*

- *$\Phi(t) < 6\kappa$ and $p_{\min}(t) > 1/\sqrt{\kappa}$.*
- *Slot t is part of an error epoch, and $p_{\min}(t) > 1/\sqrt{\kappa}$.*

In these cases, $\Phi(t)$ increases by < 2 because of packet activations during the (silent) epoch.

PROOF. Upon hearing a silent slot, all inactive packets activate. Assume that $r = M_t$ packets activate. Notice that if $p_{\min} > 1/\sqrt{\kappa}$, then $p_{\min}$ decreases when packets activate.

We first show that if $p_{\min}$ does not change due to the packet activations, then the potential does not increase due to packet activations. To show this, assume that $p_{\min}$ does not change when the packets are activated. Consider the four terms of the potential function:

- $N(t)$ does not change based on packet activations (as it is only affected by new packet injections).
- The second term of the potential function $\log C(t)$ increases by at most $4r/\ln \kappa$ because: If $c_t < c_*$, then $\log C(t) = 0$; hence we only need to consider the increase in contention above $c_* = \sqrt{\kappa}$. That is, the maximum increase in $\log C(t)$ is at most $4\kappa \log_\kappa(1 + (r/\sqrt{\kappa})/\sqrt{\kappa}) \le 4\kappa r/(\kappa \ln \kappa) = 4r/\ln \kappa$ (this inequality follows using $\ln(1 + x) \le x$).
- $s(t)$ does not increase, by assumption for this case.
- $u(t)$ decreases by $5r/\ln \kappa$ because r inactive packets became active.

We conclude that if $p_{\min}$ is not changed by the packet activations (and hence $s(t)$ is not increased), then the increase in potential due to $\log C(t)$ is less than the decrease in potential due to $u(t)$, and hence $\Phi(t)$ does not increase.

Next, we show that even if $p_{\min}$ does change due to packet activations, the potential never increases by more than 2. Consider the third term of the potential function $s(t)$. If $p_{\min}$ decreases, then $s(t)$ increases. A packet, when activated, begins with $p_j = 1/\sqrt{\kappa}$.

Thus, the most that $s(t)$ can increase is $< 4 \log_\kappa \sqrt{\kappa} = 2$ since $p_{\min} < 1$ during a silent slot. In total, summing the changes over all four parts of the potential function, the potential increases by at most: $4r/\ln \kappa + 2 - 5r/\ln \kappa < 2$.

It remains to show that $\Phi(t)$ increases (by at most 2, as already shown) only when $p_{\min}$ is reduced due to packet activations *and* either $\Phi(t) < 6\kappa$ or there is an error epoch. Consider the following three cases (immediately prior to the packet activations):

Case 1: The number of active packets $< \kappa$ and $p_{\min}$ is reduced by packet activations. If $r/\ln \kappa \ge 2$, then there is no increase in potential due to packet activations because $(4r/\ln \kappa + 2) - 5r/\ln \kappa \le 0$. Otherwise, $r < 2 \ln \kappa$.

In this latter case (where the potential might increase and $r < 2 \ln \kappa$), we can compute the potential function before the packet activations as follows:

- $N(t) \le \kappa + r$, by the case assumption. Since $r \le 2 \ln \kappa \le \kappa$, we conclude that $N(t) \le 2\kappa$.
- $\log C(t) \le 4\kappa \log_\kappa(c_t/\sqrt{\kappa}) \le 2\kappa$ because $c_t \le \kappa$.
- $s(t) \le 4 \log_\kappa(\sqrt{\kappa}) \le 2$; because $p_{\min}$ is reduced by the packet activations, we know that before these activations, it must have been the case that $p_{\min} \ge 1/\kappa$.
- $u(t) = 5r/\ln \kappa < 10$.

Thus, $\Phi(t) \le 2\kappa + 2\kappa + 2 + 10 \le 4\kappa + 12$. Given that $\kappa \ge 6$, we conclude that $\Phi(t) \le 6\kappa$. Thus, in this case, the potential increases only if $\Phi(t) \le 6\kappa$.

Case 2: The number of active packets $\ge \kappa$ and $p_{\min}$ is reduced by packet activations. Since $p_{\min}$ is reduced by the packet activations, we know that prior to their activation, $p_{\min} > 1/\sqrt{\kappa}$. It follows that $c_t > \kappa/\sqrt{\kappa} = \sqrt{\kappa}$. We know that this is a silent epoch, because otherwise the packets would not have activated. And the contention at the beginning of this silent epoch was $> \sqrt{\kappa}$. Thus, this epoch is an error epoch, by definition.

Case 3: $p_{\min}$ is *not* reduced by packet activations. As we have already shown, the potential does not increase.

Over all three cases, then, we see that potential only increases when $\Phi(t) \le 6\kappa$ and $p_{\min}$ is reduced by the packet activations, or when there is an error epoch and $p_{\min}$ is reduced by the packet activations. And in all such cases, the potential increases by at most 2. □

LEMMA 7. *A successful epoch of length ℓ, during which i packets are inserted by the adversary, will result in a decrease to the potential of at least $\ell - i(1 + 5/\ln \kappa)$.*

PROOF. First, let us look at the change in potential due to the inserted packets. The first term N_t increases by i because there are i new packets; the last term increases by $5i/\ln \kappa$ because these packets are inactive (until the next silent epoch). The other terms in the potential function are not affected by new packet injections.

Now consider the change in potential due to the successful epoch. The first term N_t will decrease by ℓ, as the epoch corresponds with a decoding window (Lemma 2) of size ℓ and hence ℓ packets are successfully delivered. The net change in the second term is non-positive, as the contention c_t does not increase

upon a successful epoch (since no broadcast probabilities are increased). Similarly, the net change in $s(t)$ is also non-positive, as $p_{\min}$ cannot decrease upon a successful epoch. (It may increase if the packet(s) with minimal joining probability are delivered.) Finally, $u(t)$ does not change since inactive packets are unaffected by a successful epoch. □

Having established preliminaries regarding the effects of packet arrivals, packet activations, and successful epochs on $\Phi(t)$, we now examine the behavior of our potential function $\Phi(t)$.

LEMMA 8. *An error epoch increases $\Phi(t)$ by at most $\kappa + i(1 + 5/\ln \kappa) + 2$, where i is the number of packets that arrive during the epoch.*

PROOF. There are two types of error epochs: a silent epoch with high contention or an overfull epoch without high contention. In both cases, the i new packets that arrive increase the first term of the potential by i and the last term of the potential by $5i/\ln \kappa$.

- When there is an overfull epoch, $p_{\min}$ decreases multiplicatively by a factor of $\kappa^{1/4}$, causing an increase to $s(t)$ of exactly 1. Changes to c_t only decrease the potential. There is no change in potential due to packet activations because the epoch is not silent.
- When there is a silent epoch, c_t increases multiplicatively by a factor of $\kappa^{1/4}$, causing an increase to $\log C(t)$ of exactly κ. Changes to $p_{\min}$ only decrease the potential. By Lemma 6, the packet activations caused by silent (error) epoch increase $\Phi(t)$ by at most 2.

Thus the overall increase to $\Phi(t)$ caused by an error epoch is at most $\kappa + 2$. □

LEMMA 9. *Consider a given non-error epoch of length ℓ, during which i packets are inserted by the adversary. If any of the following conditions hold: (i) $\Phi(t) > 6\kappa$ or (ii) $p_{\min} < 1/\sqrt{\kappa}$ or (iii) contention $\geq \kappa^{1/4}$, then $\Phi(t)$ decreases by at least $\ell(1 - 1/\kappa) - i(1 + 5/\ln \kappa)$. Otherwise, $\Phi(t)$ decreases by at least $\ell(1 - 1/\kappa) - i(1 + 5/\ln \kappa) - 2$.*

PROOF. Note that if the epoch is a successful epoch, we can immediately apply Lemma 7 to achieve the desired result. Thus we only need to consider overfull and silent epochs. Notably, if contention $\geq \kappa^{1/4}$ and $\leq \kappa^{3/4}$, then the epoch is successful (because it is a non-error epoch).

By Lemma 5, we know that the i packet arrivals will increase $\Phi(t)$ by exactly $i(1 + 5/\ln \kappa)$. By Lemma 6, we know that, if $\Phi(t) \geq 6\kappa$ or $p_{\min} \leq 1/\sqrt{\kappa}$, any packet activations that occur during this epoch do not increase $\Phi(t)$. If contention $\geq \kappa^{1/4}$, then the epoch is not silent (because it is a non-error epoch) and so no packets are activated. Otherwise, the packet activations increase $\Phi(t)$ by at most 2.

Thus, it is sufficient to show that, setting aside packet arrivals and activations, $\Phi(t)$ decreases by at least $\ell(1 - 1/\kappa)$ over this epoch. We proceed by casework, considering high contention, low contention, and contention near $c_* = \sqrt{\kappa}$.

Case 1: $c_t > \kappa^{3/4}$.
Since the epoch is not an error epoch, it cannot be a silent epoch. Thus we need only consider the case when it is an overfull epoch.

An overfull epoch is of length $\ell = \kappa$ by definition. Overfull epochs have no effect on the first and fourth terms in $\Phi(t)$. The epoch will cause a decrease to $\log C(t)$ of κ due to a multiplicative decrease in c_t by a factor of $\kappa^{1/4}$. It will also increase $s(t)$ by 1 due to $p_{\min}$ decreasing multiplicatively by a factor of $\kappa^{1/4}$. Thus we have a net decrease in Φ of $\kappa - 1 = \ell(1 - 1/\kappa)$.

Case 2: $c_t < \kappa^{1/4}$.
Since the epoch is not an error epoch, it cannot be an overfull epoch. We have already considered successful epochs, so we need only consider a silent epoch.

A silent epoch has no effect on the first and last terms in $\Phi(t)$, and the second term, $\log C(t)$, will be 0 both before and after this epoch due to contention remaining below c_*. Thus the only term affected by this epoch is $s(t)$, which will decrease by $1 = \ell$ as $p_{\min}$ increases multiplicatively by a factor of $\kappa^{1/4}$.

Case 3: $\kappa^{1/4} \leq c_t \leq \kappa^{3/4}$.
Because the epoch is not an error epoch, it must be a successful epoch. As noted earlier, we can then apply Lemma 7 to complete the proof.

□

5.3 Bounding the Potential

LEMMA 10. *Suppose that in every window of size w, for any given $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln \kappa}\right)w$ packet arrivals. Then $\Phi(t) \leq 2w$, with probability at least $1 - 1/\mathrm{poly}(w)$.*

PROOF. Consider the last time before t when Φ was less than 6κ. We label this time t'. The interval $(t', t]$ is of length $t - t'$, so at most $\left\lceil \frac{t-t'}{w} \right\rceil \left(1 - \frac{5}{\ln \kappa}\right)w < (t - t' + w)\left(1 - \frac{5}{\ln \kappa}\right)$ packets arrive during interval $(t', t]$. Furthermore, $\Phi > 6\kappa$ over this interval.

Suppose that j error epochs occur during $(t', t]$, for any arbitrary t'. By Lemma 4, we know that $j \leq \sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$, for a fixed constant c, with probability $\geq 1 - 1/\mathrm{poly}(w)$. For the remainder of the proof, we assume that this event occurs.

We observe that at least $(t - t') - j\kappa - \kappa$ time was spent in non-error epochs that complete by time t. (Note that there is potentially one incomplete epoch at the end, which has at most κ slots.)

Let ℓ be the number of slots in completed non-error epochs in the interval $(t', t]$. Let i be the number of packets that are injected during that interval. Applying Lemmas 8 and 9, it follows that:

$$\Phi(t) \leq \Phi(t') - \ell\left(1 - \frac{1}{\kappa}\right) + i\left(1 + \frac{5}{\ln \kappa}\right) + j(\kappa + 2)$$

$$\leq 6\kappa - \left(t - t' - (j+1)\kappa\right)\left(1 - \frac{1}{\kappa}\right) +$$
$$(t - t' + w)\left(1 - \frac{5}{\ln \kappa}\right)\left(1 + \frac{5}{\ln \kappa}\right) + j(\kappa + 2)$$

$$\leq 7\kappa - (t - t')\left(1 - \frac{1}{\kappa}\right) + (t - t' + w)\left(1 - \frac{25}{\ln^2 \kappa}\right) + 2j(\kappa + 2)$$

$$\leq 7\kappa + w - (t - t')\left(\frac{24}{\ln^2 \kappa}\right) + 2j(\kappa + 2).$$

Thus, if $\Phi(t) \geq 2w$ then:

$$2j(\kappa + 2) \geq w - 7\kappa + (t - t')\left(\frac{24}{\ln^2 \kappa}\right)$$

and hence:

$$j > w/(\kappa + 2) - 7 + (t - t') \left(\frac{12}{\kappa \ln^2 \kappa} \right)$$
$$\geq \sqrt{w} + (t - t') \left(\frac{12}{\kappa \ln^2 \kappa} \right).$$

This, however, contradicts our assertion that $j \leq \sqrt{w} + c(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$, for a fixed constant c, implying that $\Phi(t) < 2w$. □

THEOREM 11. *Suppose that in every window of size w, for any given $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln \kappa}\right)w$ packet arrivals. Then at any arbitrary time t, the number of packets in the system is at most $2w$ with probability at least $1 - 1/\mathrm{poly}(w)$.*

PROOF. By Lemma 10, we know that $\Phi(t) < 2w$, with probability at least $1 - 1/\mathrm{poly}(w)$. Since the first term of the potential is the number of packets in the system, this immediately implies that that are at most $2w$ packets in the system. □

6 LATENCY ANALYSIS

In this section, we consider the maximum latency of a packet. Under the assumption that the adversary can inject $(1 - o(1))w$ packets in a window of size w, it is unavoidable that the worst-case latency of a packet will be $\Omega(w)$: if the adversary injects w packets in a slot, and the maximum capacity is one packet per slot, then at least half the packets will take at least $w/2$ slots. Our goal in this section is to show that the worst-case throughput is $O(\sqrt{\kappa}w \ln^3 w)$. We also consider the batch case and show that if n packets arrive simultaneously, they will all be delivered by time $n + O(1)$.

6.1 Packet Latency Analysis

We begin by defining a ***sparse system*** event that occurs when there are very few active packets in the system. We will show that whenever a sparse system event occurs, each packet has a reasonable chance of success.

DEFINITION 3. *A **sparse system** event occurs in slot t if: (i) $\Phi(t) \leq 6\kappa$; (ii) contention $< \kappa^{1/4}$, (iii) $p_{\min} \geq 1/\sqrt{\kappa}$.*

When a sparse system event occurs, it means that every active packet was joining epochs with a relatively high probability; after the epoch completes, any new packets that activated have lowered $p_{\min}$. Notice that when a sparse system event does not occur, then the potential decreases (aside from new packets being injected): Lemma 9 shows that potential decreases in non-error epochs, and Lemma 3 shows that error epochs are unlikely.

We first show that no packet observes too many completed sparse system events before it is delivered.

LEMMA 12. *A given packet p remains active in the system for at most $O(\sqrt{\kappa} \ln w)$ sparse system events, with probability at least $1 - 1/\mathrm{poly}(w)$.*

PROOF. When a sparse system event occurs, two facts are true: (i) the probability that packet p joins that epoch is $\geq 1/\sqrt{\kappa}$ because of the condition on $p_{\min}$; (ii) the number of active packets is $\leq \kappa$. The latter fact follows because if there were more than κ active packets, then the total contention would be at least $\kappa \cdot 1/\sqrt{\kappa} \geq \sqrt{\kappa}$, contradicting the condition on contention.

Since there are at most κ active packets, then all the packets that join the epoch will be successfully delivered. Since p (once active) joins with probability $1/\sqrt{\kappa}$, we conclude that p had a $1/\sqrt{\kappa}$ probability of being successfully delivered. The probability that p fails to be delivered after $\Theta(\sqrt{\kappa} \ln w)$ sparse system events (while it is active) is at most $(1 - 1/\sqrt{\kappa})^{\Theta(\sqrt{\kappa} \ln w)} \leq 1/\mathrm{poly}(w)$. □

We now argue that, with high probability, a sparse system event occurs every $O(w \ln w)$ slots. The first step is to show that if the potential is some value α at a time t, then a sparse system event occurs within time $O((\alpha + w) \ln^2 w)$.

LEMMA 13. *Suppose that in every window of size w, for any given $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln \kappa}\right)w$ packet arrivals. Fix a time t and define $\alpha = \Phi(t)$. Then, with high probability in w, by time $t + 1.1(\alpha + w) \ln^2 w$, either all packets active at time t are delivered or a sparse system event occurs.*

PROOF. Let $t_1 = t$ and let $t_2 = t_1 + qw$ be the largest number that is congruent with $t_1 \pmod{w}$ and smaller than $t_1 + 1.1(\alpha + w) \ln^2 w$. It is necessarily true then that $t_2 - t_1$ is a multiple of w satisfying $t_2 - t_1 \geq t + (\alpha + w) \ln^2 w$ (i.e., relaxing the constant in the inequality from 1.1 to 1 allows us to take $t_2 - t_1$ to be a multiple of w). Assume, for the sake of contradiction, that no sparse system event occurs during the interval $(t_1, t_2]$. We will show that all the packets active at time t_1 are delivered with high probability.

Suppose that j error epochs occur during $(t_1, t_2]$. By Lemma 4, we know that $j \leq \sqrt{w} + 6(t_2 - t_1)/2^{\Theta(\kappa^{1/4})}$, with probability $\geq 1 - 1/\mathrm{poly}(w)$. For the remainder of the proof, we assume that this event occurs.

We observe that at least $(t_2 - t_1) - j\kappa - \kappa$ time was spent in non-error epochs that complete by time t_2. (Note that there is potentially one incomplete epoch at the end, which has at most κ slots.)

We also observe that the number of packets that are injected in the interval $(t_1, t_2]$ is at most: $\lceil \frac{t-t'}{w} \rceil \left(1 - \frac{5}{\ln \kappa}\right)w < (t_2 - t_1)\left(1 - \frac{5}{\ln \kappa}\right)$.

We can now bound the potential at time t_2. At slot t_1, $\Phi(t_1) = \alpha$; by Lemma 9 that the potential decreases by $(1 - 1/\kappa)$ in each slot that was not part of an error epoch and increases $(1 = 5/\ln \kappa)$ for each injected packet; by Lemma 8, the potential increases at most $\kappa + 2$) for each error epoch. We use this to bound how long until the potential reaches zero.

Let ℓ be the number of slots in completed non-error epochs in the interval $(t_1, t_2]$. Let i be the number of packets that are injected

during that interval. Applying Lemmas 8 and 9, it follows that:

$$\Phi(t_2) \leq \Phi(t_1) - \ell\left(1 - \frac{1}{\kappa}\right) + i\left(1 + \frac{5}{\ln\kappa}\right) + j(\kappa + 2)$$

$$\leq \alpha - (t_2 - t_1 - (j+1)\kappa)\left(1 - \frac{1}{\kappa}\right) +$$

$$(t_2 - t_1)\left(1 - \frac{5}{\ln\kappa}\right)\left(1 + \frac{5}{\ln\kappa}\right) + j(\kappa + 2)$$

$$\leq \alpha - (t_2 - t_1)\left(1 - \frac{1}{\kappa}\right) + (t_2 - t_1)\left(1 - \frac{25}{\ln^2\kappa}\right) + (2j+1)(\kappa+2)$$

$$\leq \alpha - (t_2 - t_1)\left(\frac{24}{\ln^2\kappa}\right) + 6j\kappa$$

$$\leq \alpha - (t_2 - t_1)\left(\frac{24}{\ln^2\kappa}\right) + 6\sqrt{w}\kappa + (t_2 - t_1)\frac{36\kappa}{2^{\Theta(\kappa^{1/4})}}$$

$$\leq (\alpha + 6w) - (t_2 - t_1)\left(\frac{18}{\ln^2\kappa}\right).$$

Thus, since $(t_2 - t_1) \geq (\alpha + w)\ln^2 w \geq (\alpha + w)\ln^2\kappa$, we can conclude that the potential has dropped below zero, implying that all the packets in the system have been delivered. □

A corollary of the previous lemma is that a sparse system event occurs in every interval of size $O(w\ln^2 w)$, with high probability.

COROLLARY 14. *Suppose that in every window of size w, for any given $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln\kappa}\right)w$ packet arrivals. For any interval of time (t_1, t_2) where $(t_2 - t_1) > 4w\ln^2 w$, then either every packet active at time t_1 is delivered by time t_2, or a sparse system event occurs in that interval, with probability $1 - 1/\text{poly}(w)$.*

PROOF. From Lemma 10, we know that at time t_1, $\Phi(t) \leq 2w$ with probability at least $1 - \text{poly}(w)$. From Lemma 13, we know that by time $t_1 + 1.1(\alpha + w)\ln^2 w \leq t_1 + 4w\ln^2 w \leq t_2$, either all the packets active at time t_1 are delivered or a sparse system event occurs, with probability at least $1 - \text{poly}(w)$. □

We can conclude from the preceding lemmas that, with high probability, every packet is delivered in $O(\sqrt{\kappa}w\ln^3 w)$ time.

THEOREM 15. *Suppose that in every window of size w, for any given $w \geq 16\kappa^2$, there are at most $\left(1 - \frac{5}{\ln\kappa}\right)w$ packet arrivals. Then any given packet is delivered within time $O(w\sqrt{\kappa}\ln^3 w)$ with probability $1 - 1/\text{poly}(w)$.*

PROOF. Assume that packet p is injected at time t. We first need to consider how long until packet p becomes active.

We now bound how long until a sparse system event occurs which leads to a non-error epoch; such an epoch will be silent, and hence packet p will become active. In every window of length $\Theta(w\ln^2 w)$, there is a sparse system event with probability $1 - \text{poly}(w)$ or all active packets are delivered, by Corollary 14. If all active packets are delivered, then the next epoch is necessarily silent. If a sparse system event occurs, by Lemma 3, we know that the epoch comprising the sparse system event is an error epoch with probability $1/2^{\Theta(\kappa^{1/4})} \leq 1/e$; otherwise it is silent and p becomes active. Thus, the probability that there is no silent epoch (and hence packet p remains inactive) for $\Theta(\ln w)$ intervals of length $\Theta(w\ln^2 w)$ is $\leq 1/\text{poly}(w)$.

Once p becomes active, we know by Lemma 12 that p will be delivered within $O(\sqrt{\kappa}\ln w)$ sparse system events, with probability at least $1 - \text{poly}(w)$. We know by Corollary 14 that in every interval of length $O(w\ln^2 w)$, either all packets (including p) are delivered or there is a sparse system event, with probability at least $1 - 1/\text{poly}(w)$. Thus, within $O(w\sqrt{\kappa}\ln^3 w)$, packet p is delivered with probability at least $1 - 1/\text{poly}(w)$. □

6.2 Batch Analysis

THEOREM 16. *Suppose that a batch of $n \neq 0$ packets arrives at time 0, and the adversary does not inject any packets besides this initial batch. Then with high probability in n, all of the packets have successfully been delivered by time $n(1 + 10/\kappa) + O(\kappa)$.*

PROOF. Assume n packets arrive at time 0, which is a silent epoch, and then all immediately activate. Thus at time 1 we have

$$\Phi(1) = n + \max\left(0, 4\kappa\log_\kappa\frac{n}{\kappa}\right) + 4\log_\kappa\frac{1}{1/\sqrt{\kappa}}$$

$$\leq n(1 + 5/\ln\kappa) + 2$$

We now consider $\Phi(t)$. At least $t - 1 - \kappa$ slots have been in non-error epochs since time 1, and let j be the number of error epochs. Then

$$\Phi(t) \leq \Phi(1) - (t - 1 - \kappa)(1 - 1/\kappa) + j(\kappa + 2)$$

$$\leq \Phi(1) - t(1 - 1/\kappa) + (j + 1)(\kappa + 2).$$

With high probability in n, we have, via an application of Chernoff Bounds, that $j \leq 2t/2^{\Theta(\kappa^{1/4})}$, for any $t \geq n$. We assume this holds to obtain

$$\Phi(t) \leq \Phi(1) - t(1 - 1/\kappa) + (1 + 2t/2^{\Theta(\kappa^{1/4})})(\kappa + 2).$$

Taking $t = 4\kappa + n(1 + 10/\kappa)$, we have that $\Phi(t) \leq 0$. Thus there are, with high probability in n, no packets left in the system by slot $t = 4\kappa + n(1 + 10/\kappa)$, which is $O(n(1 + 1/\kappa))$. □

REFERENCES

[1] 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. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), 1–3534.

[2] N. Abramson. 1970. The ALOHA System — Another Alternative for Computer Communications. In *Proc. of AFIPS FJCC*, Vol. 37. 281–285.

[3] N. Abramson. 1973. The Aloha system. In *Computer-Communication Networks*, N. Abramson and Frank Kuo (Eds.). Prentice-Hall, Englewood Cliffs, New Jersey, 501–518.

[4] Norman Abramson. 1977. The throughput of packet broadcasting channels. *IEEE Transactions on Communications* 25, 1 (1977), 117–128.

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

[6] D. 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.

[7] David J. Aldous. 1987. Ultimate Instability of Exponential Back-Off Protocol for Acknowledgment-Based Transmission Control of Random Access Communication Channels. *IEEE Trans. on Inform. Theory* IT-33, 2 (March 1987), 219–223.

[8] Lakshmi Anantharamu, Bogdan S Chlebus, Dariusz R Kowalski, and Mariusz A Rokicki. 2011. Medium access control for adversarial channels with jamming. In *International Colloquium on Structural Information and Communication Complexity*. Springer, 89–100.

[9] 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.

[10] Lakshmi Anantharamu, Bogdan S. Chlebus, and Mariusz A. Rokicki. 2009. Adversarial Multiple Access Channel with Individual Injection Rates. In *Proceedings of the 13th International Conference on Principles of Distributed Systems (OPODIS)*. 174–188.

[11] J.G. Andrews. 2005. Interference cancellation for cellular systems: a contemporary overview. *IEEE Wireless Communications* 12, 2 (2005), 19–29. https://doi.org/10.1109/MWC.2005.1421925

[12] Antonio Fernández Anta, Chryssis Georgiou, Dariusz R. Kowalski, and Elli Zavou. 2017. Adaptive packet scheduling over a wireless channel under constrained jamming. *Theor. Comput. Sci.* 692 (2017), 72–89. https://doi.org/10.1016/j.tcs.2017.06.020

[13] Antonio Fernández Anta, Miguel A. Mosteiro, and Jorge Ramón Muñoz. 2013. Unbounded Contention Resolution in Multiple-Access Channels. *Algorithmica* 67, 3 (2013), 295–314.

[14] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory Comput.* 8, 1 (2012), 121–164. https://doi.org/10.4086/toc.2012.v008a006

[15] 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)*. 45–54.

[16] 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)*. 325–332.

[17] Michael A. Bender, Jeremy T. Fineman, and Seth Gilbert. 2006. Contention Resolution with Heterogeneous Job Sizes. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*. 112–123. https://doi.org/10.1007/11841036_13

[18] Michael A. Bender, Jeremy T. Fineman, and Seth Gilbert. 2006. Contention Resolution with Heterogeneous Job Sizes. In *Proc. 14th Annual European Symposium on Algorithms (ESA)*. 112–123.

[19] Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Maxwell Young. 2016. How to Scale Exponential Backoff: Constant Throughput, Polylog Access Attempts, and Robustness. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 636–654.

[20] 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. https://dl.acm.org/citation.cfm?id=3276769

[21] 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)*. 105–118.

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

[23] Michael A. Bender, Tsvi Kopelowitz, Seth Pettie, and Maxwell Young. 2018. Contention Resolution with Constant Throughput and Log-Logstar Channel Accesses. *SIAM J. Comput.* 47, 5 (October 2018), 1735–1754. https://doi.org/10.1137/17M1158604

[24] Giuseppe Bianchi. 2006. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications* 18, 3 (Sept. 2006), 535–547.

[25] R. Binder, N. Abramson, F. Kuo, A. Okinaka, and D. Wax. 1975. ALOHA Packet Broadcasting: A Retrospect. In *Proceedings of the APIPS National Computer Conference*. 203–215.

[26] E. Casini, R. De Gaudenzi, and Od. R. Herrero. 2007. Contention Resolution Diversity Slotted ALOHA (CRDSA): An Enhanced Random Access Schemefor Satellite Access Packet Networks. *IEEE Transactions on Wireless Communications* 6 (2007), 1408–1419. Issue 4.

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

[28] 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*. 139–149.

[29] D. Chiu and R. Jain. 1989. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN* 17, 1 (1989), 1–14.

[30] Imrich Chlamtac and Shay Kutten. 1985. On Broadcasting in Radio Networks-Problem Analysis and Protocol Design. *IEEE Trans. Commun.* 33, 12 (1985), 1240–1246. https://doi.org/10.1109/TCOM.1985.1096245

[31] 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.

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

[33] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. 2006. Adversarial queuing on the multiple-access channel. In *Proc. Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 92–101.

[34] 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), 5.

[35] Marek Chrobak, Leszek Gasieniec, and Dariusz R. Kowalski. 2007. The Wake-Up Problem in MultiHop Radio Networks. *SIAM J. Comput.* 36, 5 (2007), 1453–1471.

[36] Gianluca De Marco, Dariusz R Kowalski, and Grzegorz Stachowiak. 2021. Deterministic Contention Resolution without Collision Detection: Throughput vs Energy. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 1009–1019.

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

[38] Mohammadreza Ebrahimi, Farshad Lahouti, and Victoria Kostina. 2020. Two-Layer Coded Channel Access With Collision Resolution: Design and Analysis. *IEEE Trans. Wirel. Commun.* 19, 12 (2020), 7986–7997. https://doi.org/10.1109/TWC.2020.3018472

[39] Martín Farach-Colton and Miguel A Mosteiro. 2007. Initializing sensor networks of non-uniform density in the weak sensor model. In *Workshop on Algorithms and Data Structures*. Springer, 565–576.

[40] Martín Farach-Colton and Miguel A Mosteiro. 2015. Initializing sensor networks of non-uniform density in the weak sensor model. *Algorithmica* 73, 1 (2015), 87–114.

[41] Chen Feng, Danilo Silva, and Frank R. Kschischang. 2013. An Algebraic Approach to Physical-Layer Network Coding. *IEEE Transactions on Information Theory* 59, 11 (2013), 7576–7596. https://doi.org/10.1109/TIT.2013.2274264

[42] Jeremy T. Fineman, Seth Gilbert, Fabian Kuhn, and Calvin Newport. 2016. Contention Resolution on a Fading Channel. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)* (Chicago, Illinois, USA) *(PODC '16)*. 155–164.

[43] Jeremy T. Fineman, Seth Gilbert, Fabian Kuhn, and Calvin Newport. 2019. Contention resolution on a fading channel. *Distributed Comput.* 32, 6 (2019), 517–533. https://doi.org/10.1007/s00446-018-0323-9

[44] Jeremy T. Fineman, Seth Gilbert, Fabian Kuhn, and Calvin C. Newport. 2016. Contention Resolution on a Fading Channel. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, George Giakkoupis (Ed.). ACM, 155–164. https://doi.org/10.1145/2933057.2933091

[45] Jeremy T. Fineman, Calvin Newport, and Tonghe Wang. 2016. Contention Resolution on Multiple Channels with Collision Detection. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing* (Chicago, Illinois, USA) *(PODC '16)*. 175–184.

[46] Mihály Geréb-Graus and Thanasis Tsantilas. 1992. Efficient Optical Communication in Parallel Computers. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 41–48.

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

[48] Leslie Ann Goldberg, Mark Jerrum, Sampath Kannan, and Mike Paterson. 2000. A Bound on the Capacity of Backoff and Acknowledgement-Based Protocols. In *Automata, Languages and Programming, 27th International Colloquium (ICALP)*. 705–716. https://doi.org/10.1007/3-540-45022-X_59

[49] Leslie Ann Goldberg, Mark Jerrum, Sampath Kannan, and Mike Paterson. 2004. A bound on the capacity of backoff and acknowledgment-based protocols. *SIAM J. Comput.* 33, 2 (2004), 313–331. https://doi.org/10.1137/S0097539700381851

[50] Leslie Ann Goldberg, Mark Jerrum, Sampath Kannan, and Mike Paterson. 2004. A bound on the capacity of backoff and acknowledgment-based protocols. *SIAM J. Comput.* 33, 2 (2004), 313–331.

[51] Leslie Ann Goldberg, Mark Jerrum, Tom Leighton, and Satish Rao. 1997. Doubly Logarithmic Communication Algorithms for Optical-Communication Parallel Computers. 26, 4 (Aug. 1997), 1100–1119.

[52] Leslie Ann Goldberg and Philip D. MacKenzie. 1999. Analysis of Practical Backoff Protocols for Contention Resolution with Multiple Servers. *J. Comput. System Sci.* 58, 1 (1999), 232–258. https://doi.org/10.1006/jcss.1998.1590

[53] 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.

[54] 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.

[55] Leslie Ann Goldberg, Yossi Matias, and Satish Rao. 1999. An Optical Simulation of Shared Memory. 28, 5 (Oct. 1999), 1829–1847.

[56] Shyamnath Gollakota and Dina Katabi. 2008. Zigzag decoding: combating hidden terminals in wireless networks. In *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Seattle, WA, USA). 159–170. https://doi.org/10.1145/1402958.1402977

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

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

[59] Albert G. Greenberg, Philippe Flajolet, and Richard E. Ladner. 1987. Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels. *J. ACM* 34, 2 (April 1987), 289–325.

[60] Albert G. Greenberg and Shmuel Winograd. 1985. A Lower Bound on the Time Needed in the Worst Case to Resolve Conflicts Deterministically in Multiple Access Channels. *J. ACM* 32, 3 (1985), 589–596.

[61] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall. 2007. Interference Cancellation: Better Receivers for a New Wireless MAC. In *Hotnets*.

[62] Johan Håstad, Frank Thomson Leighton, and Brian Rogoff. 1987. Analysis of Backoff Protocols for Multiple Access Channels (Extended Abstract). In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, Alfred V. Aho (Ed.). 241–253.

[63] Johan Håstad, Frank Thomson Leighton, and Brian Rogoff. 1996. Analysis of Backoff Protocols for Multiple Access Channels. *SIAM J. Comput.* 25, 4 (1996), 740–774. https://doi.org/10.1137/S0097539792233828

[64] Johan Hastad, Tom Leighton, and Brian Rogoff. 740-774. Analysis of Backoff Protocols for Multiple Access Channels. *SIAM J. Comput.* 25, 4 (740-774), 1996.

[65] J. Håstad, T. Leighton, and B. Rogoff. 1987. Analysis of Backoff Protocols for Multiple Access Channels. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing* (New York, New York, USA) *(STOC '87)*. Association for Computing Machinery, New York, NY, USA, 241–253. https://doi.org/10.1145/28395.28422

[66] Tomasz Jurdziński and Grzegorz Stachowiak. 2002. Probabilistic algorithms for the wakeup problem in single-hop radio networks. In *Proceedings of the International Symposium on Algorithms and Computation*. Springer, 535–549.

[67] Tomasz Jurdzinski and Grzegorz Stachowiak. 2005. Probabilistic algorithms for the wake-up problem in single-hop radio networks. *Theory of Computing Systems* 38, 3 (2005), 347–367.

[68] Tomasz Jurdzinski and Grzegorz Stachowiak. 2015. The cost of synchronizing multiple-access channels. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. 421–430.

[69] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. 2007. Embracing Wireless Interference: Analog Network Coding. *ACM SIGCOMM Computer Communication Review* 9 (02 2007).

[70] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. 2006. XORs in the air: practical wireless network coding. In *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pisa, Italy, September 11-15, 2006*, Luigi Rizzo, Thomas E. Anderson, and Nick McKeown (Eds.). ACM, 243–254. https://doi.org/10.1145/1159913.1159942

[71] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. 2008. XORs in the air: practical wireless network coding. *IEEE/ACM Trans. Netw.* 16, 3 (2008), 497–510. https://doi.org/10.1145/1399562.1399563

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

[73] Dariusz R. Kowalski. 2005. On selection problem in radio networks. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing, PODC 2005, Las Vegas, NV, USA, July 17-20, 2005*. 158–166. https://doi.org/10.1145/1073814.1073843

[74] 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.

[75] Gianluca De Marco and Dariusz R. Kowalski. 2015. Fast Nonadaptive Deterministic Algorithm for Conflict Resolution in a Dynamic Multiple-Access Channel.

[76] Gianluca De Marco and Dariusz R. Kowalski. 2017. Contention Resolution in a Non-Synchronized Multiple Access Channel. *Theoretical Computer Science* 689 (2017), 1–13.

[77] Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. 2019. Deterministic Contention Resolution on a Shared Channel. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS*. 472–482.

[78] Gianluca De Marco, Marco Pellegrini, and Giovanni Sburlati. 2007. Faster deterministic wakeup in multiple access channels. *Discret. Appl. Math.* 155, 8 (2007), 898–903.

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

[80] Jeannine Mosely and Pierre Humblet. 1985. A class of efficient contention resolution algorithms for multiple access channels. *IEEE transactions on Communications* 33, 2 (1985), 145–151.

[81] 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.

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

[83] Prabhakar Raghavan and Eli Upfal. 1995. Stochastic contention resolution with short delays. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. 229–237.

[84] Prabhakar Raghavan and Eli Upfal. 1998. Stochastic Contention Resolution With Short Delays. *SIAM J. Comput.* 28, 2 (1998), 709–719.

[85] Prabhakar Raghavan and Eli Upfal. 1999. Stochastic Contention Resolution With Short Delays. 28, 2 (April 1999), 709–719.

[86] 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)*. 179–193.

[87] 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 (ICDCS)*. 507–516.

[88] 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)*. 291–300.

[89] Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2013. Competitive Throughput in Multi-Hop Wireless Networks Despite Adaptive Jamming. *Distributed Computing* 26, 3 (2013), 159–171.

[90] Andréa W. Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. 2013. An Efficient and Fair MAC Protocol Robust to Reactive Interference. *IEEE/ACM Trans. Netw.* 21, 3 (2013), 760–771. https://doi.org/10.1109/TNET.2012.2210241

[91] Lawrence G. Roberts. 1975. ALOHA packet system with and without slots and capture. *SIGCOMM Compuer Communications Review* 5, 2 (1975), 28–42.

[92] John F Shoch and Jon A Hupp. 1980. Measured performance of an Ethernet local network. *Commun. ACM* 23, 12 (1980), 711–721.

[93] 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).

[94] Arash Saber Tehrani, Alexandros G. Dimakis, and Michael J. Neely. 2011. SigSag: Iterative detection through soft message-passing. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*. IEEE, 1017–1025. https://doi.org/10.1109/INFCOM.2011.5934874

[95] David Tse and Pramod Viswanath. 2005. *Fundamentals of Wireless Communication*. Cambridge University Press.

[96] B. S. Tsybakov and N. B. Likhanov. 1987. Upper Bound on the Capacity of a Random Multiple-Access System. *Problemy Peredachi Informatsii* 23, 3 (1987), 64–78.

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

[98] 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.

SIAM J. Comput. 44, 3 (2015), 868–888.