Softening the Impact of Collisions in Contention Resolution

Umesh Biswas[1], Trisha Chakraborty[2], and Maxwell Young[1]

[1] Department of Computer Science and Engineering, Mississippi State University, MS 39762, USA
ucb5@msstate.edu, myoung@cse.msstate.edu
[2] Amazon Web Services, MN 55401, USA
trichakr@amazon.com

Abstract. Contention resolution addresses the problem of coordinating access to a shared communication channel. Time is discretized into synchronized slots, and a packet can be sent in any slot. If no packet is sent, then the slot is empty; if a single packet is sent, then it is successful; and when multiple packets are sent at the same time, a collision occurs, resulting in the failure of the corresponding transmissions. In each slot, every packet receives ternary channel feedback indicating whether the current slot is empty, successful, or a collision.

Much of the prior work on contention resolution has focused on optimizing the makespan, which is the number of slots required for all packets to succeed. However, in many modern systems, collisions are also costly in terms of the time they incur, which existing contention-resolution algorithms do not address.

In this paper, we design and analyze a randomized algorithm, COLLISION-AVERSION BACKOFF (CAB), that optimizes both the makespan and the collision cost. We consider the static case where an unknown $n \geq 2$ packets are initially present in the system, and each collision has a known cost $\mathcal{C}$, where $1 \leq \mathcal{C} \leq n^{\kappa}$ for a known constant $\kappa \geq 0$. With error probability polynomially small in n, CAB guarantees that all packets succeed with makespan and a total expected collision cost of $\tilde{O}(n\sqrt{\mathcal{C}})$. We give a lower bound for the class of fair algorithms: where, in each slot, every packet executing the fair algorithm sends with the same probability (and the probability may change from slot to slot). Our lower bound is asymptotically tight up to a $\texttt{poly}(\log n)$-factor for sufficiently large $\mathcal{C}$.

***Keywords*—** Distributed computing, contention resolution, collision cost.

1 Introduction

Contention resolution addresses the fundamental challenge of coordinating access by devices to a shared resource, which is typically modeled as a multiple access channel. Introduced with the development of ALOHA in the early 1970s [1], the problem of contention resolution remains relevant in WiFi networks [16], cellular networks [32], and shared-memory systems [8, 24].

Our problem is described as follows. There are $\boldsymbol{n} \geq 2$ devices present at the start of the system, each with a packet to send on the shared channel, and n is *a priori* unknown. For ease of presentation, we adopt a slight abuse of language by referring to packets sending themselves (rather than referring to devices that send packets). Time proceeds in discrete, synchronized ***slots***, and each slot has size that can accommodate the transmission of a packet. For any fixed slot, the channel provides ternary feedback, allowing each packet to learn whether 0 packets were sent, 1 packet was sent (a ***success***), or $2+$ packets were sent (a ***collision***). Sending on the channel is performed in a distributed fashion; that is, *a priori* there is no central scheduler or coordinator. Traditionally, the contention-resolution problem focuses on minimizing the number of slots until all n packets succeed, which is referred to as the ***makespan***.

However, in many modern systems, collisions also have a significant impact on performance. In the ***standard cost model***, each collision can be viewed as a wasted slot that increases the makespan by 1, since no packet can succeed. Yet, in many settings, a collision wastes *more* than a single slot, and we denote this cost by $\boldsymbol{\mathcal{C}}$.[3] This cost can vary widely across different systems. In intermittently-connected mobile wireless networks [36], failed communication due to a collision may result in the sender having to wait until the intended receiver is again within transmission range. For WiFi networks, a collision may consume time proportional to the packet size [6]. In shared memory systems, concurrent access to the same memory location can result in delay proportional to the number of contending processes [8]. Thus, makespan gives only part of the performance picture, since collisions may add to the time until all packets succeed.

[3] We may also view $\mathcal{C}$ as an upper bound on the cost of a collision if there is some variance.

In this work, we account for the cost of collisions, in addition to the makespan. As we discuss later (see Section 3), existing contention-resolution algorithms do not perform well in this setting. For instance, randomized binary exponential backoff (BEB) [30], arguably the most popular of all contention-resolution algorithms, incurs a collision cost of $\Omega(n\,\mathcal{C})$, where $\mathcal{C}$ is the cost of a single collision. New ideas are needed to achieve better performance in this model.

1.1 Model and Notation

Our work addresses the case where an $n \geq 2$ number of packets are active at the start of the system. There is no known upper bound on n *a priori*, and packets do not have identifiers. A packet is ***active*** if it is executing a protocol; otherwise, the packet has terminated. Each packet must be successfully sent in a distributed fashion (i.e., there is no scheduler or central authority) on a multiple access channel, which we now describe.

Multiple Access Channel. Time is divided into disjoint ***slots***, where each slot is sized to accommodate a single packet. In any slot, a packet may send itself or it may listen. For any fixed slot, if no packet is sent, then the slot is ***empty***; if a single packet is sent, then it is ***successful***; and if two or more packets are sent, then there it is a ***collision*** and none of these packets is successful. A packet that transmits in a slot immediately learns of success or failure; if it succeeds, the packet terminates immediately. Any packet that is listening to a slot, learns which of these three cases occurred; this is the standard ***ternary feedback model***. Likewise, for any slot, a packet that is sending can also determine the channel feedback; either the packet succeeded (which it learns in the slot) or it failed (from which the packet infers a collision occurred). Packets cannot send messages to each other, and no other feedback is provided by the channel.

Metrics. A well-known measure of performance in prior work is the ***makespan***, which is the number of slots required for all n packets to succeed. An equivalent metric in the static case is, ***throughput***, which is defined as the time for n successes divided by the makespan.

In our model, each collision incurs a known cost of $\boldsymbol{\mathcal{C}}$, where $1 \leq \mathcal{C} \leq n^{\kappa}$, for a known constant $\boldsymbol{\kappa} \geq 0$.[4] Given a contention-resolution algorithm, the pertinent costs are: (i) the ***collision cost***, which is the number of collisions multiplied by $\mathcal{C}$, and (ii) the makespan. Often we refer to "cost", by which we mean the maximum of (i) and (ii), unless specified otherwise.

Notation. Throughout this manuscript, $\mathbf{lg(\cdot)}$ refers to logarithm base 2, and $\mathbf{ln(\cdot)}$ refers to the natural logarithm. We use $\mathbf{\log^{y}(x)}$ to denote $(\log(x))^{y}$, and we use $\mathbf{poly(x)}$ to denote x^{y} for some constant $y \geq 1$. The notation $\boldsymbol{\tilde{O}}$ denotes the omission of a $\mathtt{poly}(\log n)$ factor. We say an event occurs ***with high probability (w.h.p.) in n*** (or simply "with high probability") if it occurs with probability at least $1 - O(1/n^{c})$, for any tunable constant $c \geq 1$.

For any random variable X, we say that w.h.p. the expected value $E[X] \leq x$ if, when tallying the events in the expected value calculation, the sum total probability of those events where $X > x$ is $O(1/n^{c})$ for any tunable constant $c \geq 1$. In our analysis, the expected collision cost holds so long as the subroutine DIAGNOSIS (described later) gives correct feedback to the packets, and this occurs with high probability.

1.2 Our Results

We design and analyze a new algorithm, **COLLISION-AVERSION BACKOFF *(CAB)***, that solves the static contention resolution problem with high-probability bounds on makespan and the expected collision cost. The following theorems provide our formal result.

Theorem 1. *With high probability in n,* COLLISION-AVERSION BACKOFF *guarantees that all n packets succeed with a makespan of $\tilde{O}(n\sqrt{\mathcal{C}})$, and an expected collision cost of $\tilde{O}(n\sqrt{\mathcal{C}})$.*

How does this compare with prior results? For starters, consider SAWTOOTH-BACKOFF (STB), which w.h.p. has an asymptotically-optimal makespan of $\Theta(n)$, but incurs $\Omega(n)$ collisions. The expected cost for CAB is superior to STB when $\mathcal{C}$ is at least polylogarithmic in n. In Section 1.3, we elaborate on how our result fits with previous work on contention resolution.

In a well-known lower bound in the standard cost model, Willard [35] defines and argues about ***fair*** algorithms. These are algorithms where, in a fixed slot, every active packet sends with the same probability (and the probability may change from slot to slot). Our lower bound applies to fair algorithms as follows.

[4] Note that knowing $\mathcal{C}$ and κ implies only a lower (not upper) bound on n.

Theorem 2. *Any fair algorithm that w.h.p. has a makespan of $\tilde{O}(n\sqrt{\mathcal{C}})$ has w.h.p. an expected collision cost of $\tilde{\Omega}(n\sqrt{\mathcal{C}})$ for $\mathcal{C} = \Omega(n^2)$.*

Since CAB is fair and guarantees w.h.p. a makespan of $\tilde{O}(n\sqrt{\mathcal{C}})$, our upper bound is asymptotically tight up to a $\texttt{poly}(\log n)$-factor for sufficiently large $\mathcal{C}$. A more general form of our lower bound is given in Section 4, along with additional discussion.

1.3 Why Care About Collision Costs?

Here, we discuss the potential value of our result and exploring beyond the standard cost model for contention resolution.

A Simple Answer. Prior contention-resolution algorithms that optimize for makespan suffer from many collisions. For example, arguably the most famous backoff algorithm is BEB (despite its sub-optimal makespan [9]), which has $\Omega(n)$ collisions. Similarly, STB has asymptotically-optimal makespan, but also suffers $\Omega(n)$ collisions [6]. Thus, these algorithms have cost $\tilde{\Theta}(n + n\mathcal{C})$. In contrast, the expected completion time for CAB is $\tilde{O}(n\sqrt{\mathcal{C}})$, which is superior whenever $\mathcal{C} = \texttt{poly}(\log n)$.

In the extreme, even a hypothetical algorithm that suffers (say, w.h.p.) a single collision would do poorly by comparison if collisions are sufficiently costly. Specifically, such an algorithm pays $\mathcal{C}$, which is asymptotically worse than the expected completion time of CAB for $\mathcal{C} = \tilde{\omega}(n^2)$.

Perhaps the above is reasonable motivation (we think it is), but why not also consider the cost of successes?[5] Below, we argue that reducing collision cost remains important, and that adding a non-unit cost for successes does not seem interesting from a theory perspective.

Connecting to the Standard Cost Model. Let us *temporarily* consider the implications of a cost model where a success and a collision each have cost $P \geq 1$. In the standard cost model $P = 1$. In WiFi networks, both costs are also roughly equal, where $P \gg 1$ [6].

In this context, reconsider STB's performance. Since, n packets must succeed, a cost of at least nP is unavoidable, and the additional $\Theta(nP)$ cost from collisions becomes (asymptotically) unimportant. (Indeed, any algorithm must pay at least nP for successes; given this unavoidable cost, it does not seem interesting from a theory perspective to consider a model where a success costs far more than a collision, since the cost from successes would dominate.) Likewise, CAB must also pay at least nP, which is (asymptotically) no better than STB.

Does reducing collision costs matter here? Yes, and the value of our result is best viewed via throughput (recall Section 1.1). STB has cn collisions, for some constant $c > 0$, and (ignoring empty slots) its throughput is less than $nP/(nP + (cn)P) < 1/(1+c) < 1 - 1/c$; that is, the throughput is bounded away from 1 by a constant amount. Doing a similar calculation for CAB, there is nP cost for n packets to succeed, plus $O(n\sqrt{P})$, which accounts for collisions and empty slots. Thus, the expected throughput is $E[nP/T]$ where T is the completion time. Since, $E[1/T] \geq 1/E[T]$, we have $E[nP/T] \geq nP/(nP + \tilde{O}(n\sqrt{P})) > 1 - \tilde{O}(1/\sqrt{P})$.

Thus, for the standard cost model, with $P = 1$, our result is (unsurprisingly) not an improvement. However, as P grows, our result provides better throughput in expectation, approaching 1. There are settings where we expect P to be large, such as: wireless networks where P can be commensurate with packet size, routing in mobile networks where communication failure increases latency [36], and shared memory systems where concurrent access to the same memory location by multiple processes results in delay [8].

2 Related Work

There is a large body of work on the static case for contention resolution, where n packets arrive together and initiate the contention resolution algorithm; this is often referred to as the ***static*** or ***batched-arrival*** setting. Bender et al. [9] analyze the makespan for BEB, as well as other backoff algorithms; surprisingly, they show that BEB has sub-optimal makespan.

An optimal backoff algorithm is STB [26, 28]. Since we borrow from STB to create one of our subroutines (discussed further in Section 3.1), so we describe it here. Informally, STB works by executing over a doubly-nested loop, where the outer loop sets the current window size w to be double the one used in the preceding outer loop. Additionally, for

[5] Empty slots do not seem to warrant such consideration since, by definition, nothing is happening in such slots, and so a per-cost of 1 makes sense.

each such window, the inner loop executes over a ***run*** of $\lg w$ windows of decreasing size: $w, w/2, w/4, ..., 1$. For each window, every packet chooses a slot uniformly at random (u.a.r.) to send in.

The static setting has drawn attention from other angles. Bender et al. [10] examines a model where packets have different sizes under the binary feedback model. Anderton et al. [6] provide experimental results and argue that packet size should be incorporated into the definition of makespan, since collisions tend to cost time proportional to packet size.

To the best of our knowledge, our work is the first to propose an algorithm for minimizing collision cost, and so it makes sense to start with the static setting. However, we note the dynamic setting has been addressed (under the standard cost model), where packet arrival times are governed by a stochastic process (see the survey by Chlebus [20]). Another direction that the research community has pursued is the application of adversarial queueing theory (AQT) [17] to contention resolution; for examples, see [22, 21, 32]. Under the AQT setting, packet arrival times are dictated by an adversary, but typically subject to constraints on the rate of packet injection and how closely in time (how bursty) these arrivals may be scheduled. Even more challenging, there is a growing literature addressing the case where packet arrival times are set by an unconstrained adversary; see [23, 14, 25]. Recent work in this area addresses additional challenges facing modern networks, such as energy efficiency [29, 15], malicious disruption of the shared channel [12, 7, 31, 19, 5, 4], and the ability to have a limited number of concurrent transmission succeed [13].

3 Technical Overview for Upper Bound

Due to space constraints, our proofs for the upper bound are provided in Section A of our appendix. However, we do reference some well-known inequalities based on the Taylor series that are omitted here, but can be found in Section A.1.

Here, we present an overview of our analysis, with the aim of imparting some intuition the design choices behind CAB, as well as highlighting the novelty of our approach. To this end, we first consider two natural—but ultimately flawed—ideas for solving our problem.

Straw Man 1. An immediate question is: *Why can we not use a prior contention-resolution algorithm to solve our problem?* For example, in the static setting, a well-known backoff algorithm, such BEB guarantees w.h.p. that all packets succeed with makespan $\Theta(n \log n)$ [9]. Under BEB, the packets execute over a sequence of disjoint windows, where window $i \geq 0$ consists of 2^i contiguous slots. Every active packet sends in a slot chosen u.a.r. from the current window. Unfortunately, a constant fraction of slots in each window $i \leq \lg(n) + O(1)$ will be collisions. Each collision imposes a cost of $\mathcal{C}$ leading to a collision cost of $\Omega(n\mathcal{C})$. □

Despite yielding a poor result, this straw man provides three useful insights. First, we cannot have packets start with a "small" window, since this leads to many collisions. Many backoff algorithms start with a small window, and will not yield good performance for this same reason.

Second, we should seek to better (asymptotically) balance the costs of makespan and collisions. Under BEB, these costs are highly unbalanced, being $\Theta(n \log n)$ and $\Omega(n\mathcal{C})$, respectively. We may trade off between makespan and collision cost; that is, we can make our windows larger, which increases our makespan, in order to dilute the probability of collisions.

Third, a window of size $\Theta(n\sqrt{\mathcal{C}})$ seems to align with these first two insights; that is, it appears to asymptotically balance makspan and collision cost. To understand the latter claim, suppose that in each slot, every packet sends with probability $\Theta(1/(n\sqrt{\mathcal{C}}))$. We can argue (informally) that, for any fixed slot, the probability of any two packets colliding is at least $\binom{n}{2}\Theta(1/n\sqrt{\mathcal{C}})^2 = O(1/\mathcal{C})$. Thus, over $n\sqrt{\mathcal{C}}$ slots in the window, the expected number of collisions is $O(n\sqrt{\mathcal{C}}/\mathcal{C}) = O(n/\sqrt{\mathcal{C}})$, and each collision has cost $\mathcal{C}$, so the expected collision cost is $O(n\sqrt{\mathcal{C}})$. Of course, this informal analysis falls short, since collisions may involve more than two packets, and we have not shown that all n packets can succeed over this single window (they cannot). Yet, this insight offers us hope that we can outperform prior backoff algorithms by achieving costs that are $o(n\mathcal{C})$.

How should we find a window of $\Theta(n\sqrt{\mathcal{C}})$? Since n is unknown *a priori*, we cannot simply instruct packets to start with that window size. It is also clear from the above discussion that we cannot grow the window to this size using prior backoff algorithms. This obstacle leads us to our second straw man.

Straw Man 2. Perhaps we can estimate n and then start directly with a window of size $\Theta(n\sqrt{\mathcal{C}})$. A well-known "folklore" algorithm for estimating n is the following. In each slot $i \geq 0$, each packet sends with probability $1/2^i$; otherwise, the packet listens. This algorithm, along with improvements to the quality of the estimation, is explored by Jurdzinski et al. [29], but we sketch why it works here.

Intuitively, when i is small, say a constant, then the probability of an empty slot is very small: $(1 - 1/2^{\Theta(1)})^n \leq e^{-\Theta(n)}$ by Fact 1(a). However, once $i = \lg(n)$, then the probability of an empty slot is a constant: $(1 - 1/2^{\lg(n)})^n =$

$(1-1/n)^n \geq e^{\Theta(1)}$ by Fact 1(c). In other words, once a packet witnesses an empty slot, it can infer that its sending probability is $\Theta(1/n)$, and thus the reciprocal yields an estimate of n to within a constant factor.

At first glance, it may seem that we can estimate n, and then proceed to send packets in a window of size $\Theta(n\sqrt{\mathcal{C}})$. Unfortunately, for each slot $i \leq \lg(n)$, this algorithm (and others like it) will likely incur a collision, and thus the expected collision cost is $\Omega(\mathcal{C}\log n)$. To see why this is a problem, suppose that $\mathcal{C} = n^4$. This implies that the expected collision cost is $\tilde{\Omega}(\mathcal{C}) = \tilde{\Omega}(n^4)$, while the makespan is at best $O(n\sqrt{\mathcal{C}}) = O(n\sqrt{n^4}) = O(n^3)$. That is, there is an n-factor discrepancy between the expected collision cost and makespan, which grows worse for larger values of $\mathcal{C}$; this approach is not trading off well between these two metrics. □

Even though the above algorithm racks up a large collision cost, it highlights how channel feedback can provide useful hints. If packets are less aggressive with their sending, we can reduce collisions while still receiving feedback that lets us reach our desired window size of $\Theta(n\sqrt{\mathcal{C}})$.

3.1 Our Approach

Motivating Sample Size. Hoping to avoid the problems illustrated in our discussion above, all packets begin with an initial current window size that can be "large"; the exact size we motivate momentarily. Recall that we aim for packets to tune their sending probability to be $\Theta(1/(n\sqrt{\mathcal{C}}))$, so we are aiming for a window size of $\Theta(n\sqrt{\mathcal{C}})$. However, this must be achieved with low expected collision cost. To do this, the packets execute over a ***sample*** of s slots. For each slot in the sample, every packet sends with probability 1 divided by the current window size, and monitors the channel feedback.

What does this channel feedback from the sample tell us? Suppose we have reached our desired window size of $\Theta(n\sqrt{\mathcal{C}})$. Then, the window size may be large relative to n, and so we should expect empty slots to occur frequently, while successes *do* occur, but less often; the probability of success is approximately $\binom{n}{1}(1/(n\sqrt{\mathcal{C}})) \approx 1/\sqrt{\mathcal{C}}$. To correctly diagnose w.h.p. (using a Chernoff bound) that $w = \Theta(n\sqrt{\mathcal{C}})$, we rely on receiving $\Theta(\log n)$ successes. This dictates our sample size to be $s = \Theta(\sqrt{\mathcal{C}}\log(n\sqrt{\mathcal{C}})) = \Theta(\sqrt{\mathcal{C}}\log(n))$, since $\mathcal{C} \leq n^\kappa$.

The details behind this intuition are given in Section A.2 of our appendix, where we show that samples of size $\Theta(\sqrt{\mathcal{C}}\log(n))$ are sufficient to make correct decisions that tune the window size to be $\Theta(n\sqrt{\mathcal{C}})$. Furthermore, in Section A.3 of our appendix, we show that the corresponding cost from this tuning is $\tilde{O}(n\sqrt{\mathcal{C}})$.

Motivating Initial Window Size. Suppose we are not at our desired window size. Then, the feedback from sampling tells us what to do. If our current window size is too large (i.e., our sending probability is too low), then the number of successes will be "small", since most slots are empty, and we should decrease our window size. Else, if the current window size is too small, (i.e., our sending probability is too high) then the number of successes is again "small", since most slots are collisions, and we should increase the window size. But wait, in the latter case, can we tolerate the cost of the resulting collisions? In the worse case, the entire sample might consist of collisions, resulting in a potentially large cost of $s\,\mathcal{C} = \Theta(\mathcal{C}^{3/2}\log n)$.

We remedy this problem by setting our initial window size to be $\mathcal{C}$. Then, the probability of a collision in a fixed slot is approximately $\binom{n}{2}(1/\mathcal{C}^2)$ and so the expected cost is roughly $T = \binom{n}{2}(1/\mathcal{C}^2)\,\mathcal{C} = \Theta(n^2/\mathcal{C})$. Reasonaing informally, if $\mathcal{C} > n$, then $T = O(n)$, and over the sample the expected cost is $O(n\sqrt{\mathcal{C}}\log(n))$, as desired. Else, if $\mathcal{C} \leq n$, then even if our sample does consist entirely of collisions, the resulting cost is $\Theta(\mathcal{C}\sqrt{\mathcal{C}}\log n) = O(n\sqrt{\mathcal{C}}\log n)$. This reasoning is formalized in Lemma 13 of our appendix.

In light of the above, we emphasize that CAB does not avoid all collisions; in fact, many collisions may occur when the collision cost is "small" (i.e., $\mathcal{C} \leq n$). However, as the collision cost grows "larger" (i.e., $\mathcal{C} > n$), CAB expresses an increasing aversion to collisions by having packets tune their respective sending probabilities such that we expect $o(1)$ collisions per sample. The cost analysis for sampling is given in Section A.3 of our appendix.

Borrowing from Sawtooth Backoff. By using sampling, ultimately a window size of $\Theta(n\sqrt{\mathcal{C}})$ is reached. At this point, every active packet executes the analog of the final run of STB (recall Section 2). Specifically, each packet sends with probability $p = \Theta(1/(n\sqrt{\mathcal{C}}))$, which we show is sufficient to have at least half of the active packets succeed (see Lemma 20). Then, the window is halved, and the process repeats where the remaining packets send with probability $p/2$. This halving process continues until all packets succeed. By the sum of a geometric series, the number of slots in this process is $O(n\sqrt{\mathcal{C}})$. Informally, the expected collision cost per slot is roughly $\binom{n}{2}(1/(n\sqrt{\mathcal{C}})^2)\mathcal{C} = O(1)$, and thus $O(n\sqrt{\mathcal{C}})$ over all slots in the window. The analysis of cost is provided in Section A.4 of our appendix.

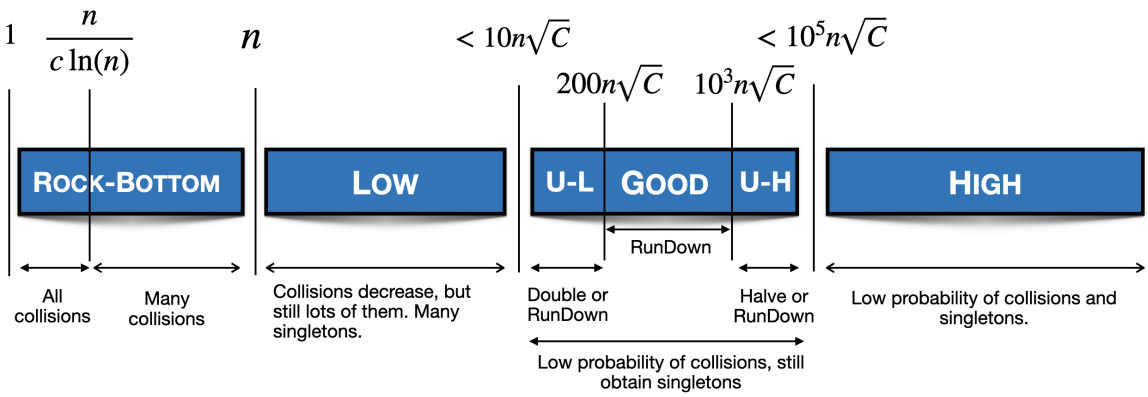

Fig. 1: Illustration of the ranges discussed in Section 3.2.

3.2 Our Algorithm and Overview of Analysis

This section describes and gives intuition for CAB, whose pseudocode is given in Figure 2. As stated in our model (Section 1.1), when a packet succeeds it terminates immediate; for ease of presentation, we omit this from our pseudocode.

From a high-level view, each packet keeps track of a current window size, w_{cur}; this size is critical, as it dictates the per-slot sending probability of each packet, which is $\Theta(1/w_{\text{cur}})$. Each packet keeps track of its own notion of a current window. However, in each slot, since every packet is either listening or sending (and, thus, learning whether the slot contained a success or a collision), all packets receive the same channel feedback and adjust their respective current window identically (stated formally in Lemma 8). Therefore, for ease of presentation, we refer only to a single current window.

Defining Ranges. In order to describe how CAB works, we define the six size ***ranges*** that the current window (or just "window" for short), w_{cur}, can belong to during an execution.

- ROCK-BOTTOM: $[1, n)$.
- LOW: $[n, 10n\sqrt{C})$.
- UNCERTAIN-LOW: $[10n\sqrt{C}, 200n\sqrt{C})$.
- GOOD: $[200n\sqrt{C}, 10^3 n\sqrt{C})$.
- UNCERTAIN-HIGH: $[10^3 n\sqrt{C}, 10^5 n\sqrt{C})$.
- HIGH: $[10^5 n\sqrt{C}, \infty)$.

These ranges are depicted in Figure 1. We note that the particular constants in these ranges are not special; they are chosen for ease of analysis. To gain intuition, we now describe the events we expect to witness in the ROCK-BOTTOM, LOW, GOOD, and HIGH ranges when CAB executes; we defer an in-depth discussion of UNCERTAIN-LOW and UNCERTAIN-HIGH until the end of the next subsection.

The ROCK-BOTTOM range captures "tiny" window sizes, starting from 1 up to $n - 1$. In this range, the probability of sending exceeds $1/n$, and we expect that most slots will be collisions. The next range is LOW, and it includes window sizes from n to just below $10n\sqrt{C}$. This range represents a moderate increase in window size, allowing for more successes than ROCK-BOTTOM, although there can still be many collisions. As discussed in Section 3.1, in ROCK-BOTTOM and in the bottom portion of LOW, we can afford to have an single sample consist entirely of collisions, since $\mathcal{C} = O(n)$ (see Lemma 16). However, lingering in these ranges would ultimately lead to sub-optimal expected collision cost.

The GOOD range spans from $200n\sqrt{C}$ to just below $10^3 n\sqrt{C}$. In this range, the window sizes are sufficiently large that we expect $o(1)$ collisions, along with handful of successes; notably, less successes than LOW. This turns out to be a "good" operating range for the algorithm, where the balance between collision costs and makespan is achieved, as discussed in Section 3 (i.e., our third insight after Straw Man 1).

The HIGH range covers window sizes from $10^5 n\sqrt{C}$ and above. In this range, the window sizes are very large, leading to a very low probability of collisions, which is good. However, there are also far-fewer successes compared to GOOD, which means that lingering in this range would lead to a sub-optimal number of slots until all packets succeed.

COLLISION-AVERSION BACKOFF

Initial window size $w_{cur} \leftarrow \mathcal{C}$

repeat
  #successes, #collisions $\leftarrow 0$
  COLLECT-SAMPLE($\mathcal{C}, w_{\text{cur}}$)
  DIAGNOSIS(#successes, #collisions, w_{cur})
until *true*;

Function COLLECT-SAMPLE*($\mathcal{C}, w_{cur}$)*:
  #successes $\leftarrow 0$
  #collision $\leftarrow 0$
  for *slot* $i = 1$ **to** $d\sqrt{\mathcal{C}}\ln(w_{cur})$ **do**
    Send with probability $1/w_{\text{cur}}$; otherwise, listen
    if *slot is a success* **then**
      #successes++
    else if *slot is a collision* **then**
      #collisions++

Function DIAGNOSIS*(#successes, #collisions, w_{cur})*:
  if *#successes* $> \frac{2d\ln(w_{cur})}{10^5}$ **then**
    if *#successes* $\leq \frac{d\ln(w_{cur})}{20e}$ **then**
      if *#collisions* $\geq \frac{d\sqrt{\mathcal{C}}\ln(w_{cur})}{8e^2}$ **then**
        $w_{\text{cur}} \leftarrow 2w_{\text{cur}}$
      else
        Execute RUNDOWN (w_{cur})
    else
      $w_{\text{cur}} \leftarrow 2w_{\text{cur}}$
  else
    if *#collisions* $\geq \frac{d\sqrt{\mathcal{C}}\ln(w_{cur})}{8e^2}$ **then**
      $w_{\text{cur}} \leftarrow 2w_{\text{cur}}$
    else
      $w_{\text{cur}} \leftarrow w_{\text{cur}}/2$

Function RUNDOWN*(w_{cur})*:
  $w_0 \leftarrow w_{\text{cur}}$
  while $w_{cur} \geq 8\sqrt{\mathcal{C}}\lg(w_0)$ **do**
    for *each slot* $j = 1$ **to** w_{cur} **do**
      Send packet with probability $2/w_{\text{cur}}$
    $w_{\text{cur}} \leftarrow w_{\text{cur}}/2$
  for $i = 1$ **to** $c\ln(w_0)$ **do**
    for *each slot* $j = 1$ **to** w_0 **do**
      Send packet with probability $2/w_0$

Fig. 2: Pseudocode for COLLISION-AVERSION BACKOFF.

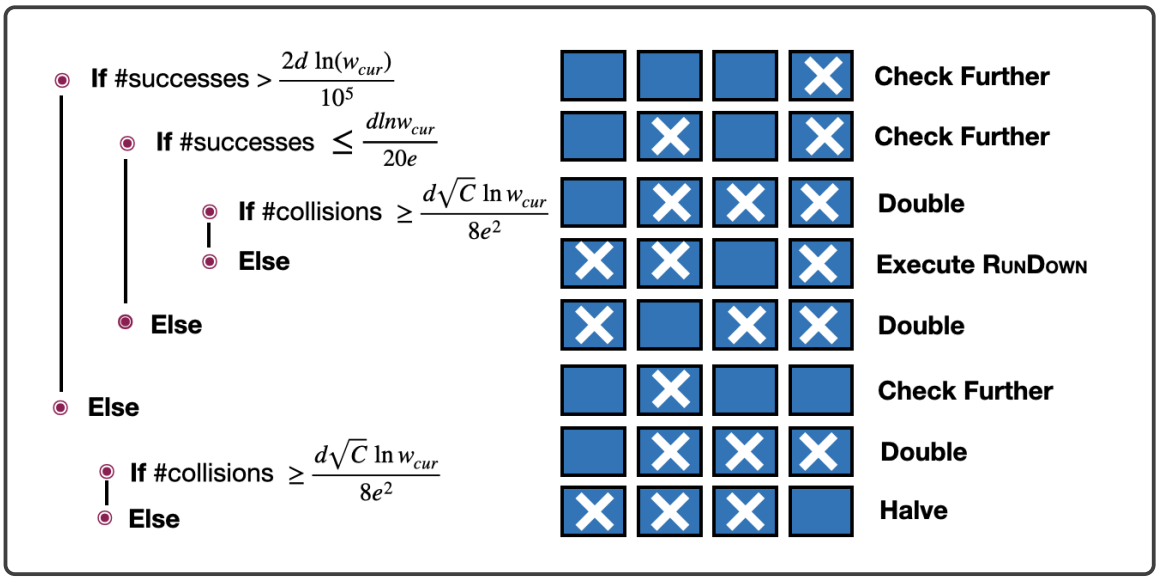

Fig. 3: The elimination of ranges as discussed in Section 3.2.

The UNCERTAIN-LOW and UNCERTAIN-HIGH ranges span $[10n\sqrt{C}, 200n\sqrt{C})$ and $[10^3 n\sqrt{C}, 10^5 n\sqrt{C})$, respectively. Why do we have these ranges? They capture values of w_{cur} where we cannot know w.h.p. exactly what will happen, although we *do* prove that the algorithm is making "progress". We will elaborate on this further after describing our methods for sampling and interpreting channel feedback, which we address next.

Sampling and Diagnosing Feedback. CAB navigates these ranges by using two subroutines: **COLLECT-SAMPLE** and **DIAGNOSIS**. As motivated earlier in Section 3.1, a sample is a contiguous set of $d\sqrt{C}\ln(w_{\text{cur}})$ slots that are used by each packet to collected channel feedback. Specifically, under COLLECT-SAMPLE, in each slot of the sample, every packet sends independently with probability $1/w_{\text{cur}}$ and records the result if it is a success or a collision.

Based on the number of successes and the number of collisions, DIAGNOSIS attempts to determine the range to which w_{cur} belongs. We discuss these thresholds in order to give some intuition for how DIAGNOSIS is making this determination. To simplify the presentation, we omit discussion of UNCERTAIN-LOW and UNCERTAIN-HIGH until the end of this section. The process by which CAB homes in on the range to which w_{cur} belongs is depicted in Figure 3.

We begin with the first if-statement on Line 17. This line checks if the number of successes is at least a logarithmic amount (in w_{cur}); if so, this indicates that w_{cur} cannot be in the HIGH range, which would yield far fewer successes (see Lemma 13). Therefore, if meet the conditional on Line 17, it must be the case that w_{cur} belongs to ROCK-BOTTOM, LOW, or GOOD.

Line 18 checks whether the number of successes falls below a "large" logarithmic amount; if not, then we are seeing a "large" number of successes that indicates w_{cur} cannot be in ROCK-BOTTOM or in GOOD, and so must be in LOW (see Lemma 10). Otherwise, the number of successes falls below this logarithmic amount, indicating that w_{cur} is in the ROCK-BOTTOM or in GOOD ranges. To discern between these two cases, Line 19 checks if the number of collisions exceeds $\frac{d\sqrt{C}\ln(w_{\text{cur}})}{8e^2}$, which indicates $w_{\text{cur}} \in$ ROCK-BOTTOM (see Lemmas 6, 7, and 9). Otherwise, w_{cur} falls within the GOOD and the job of DIAGNOSIS is complete — at this point, RUNDOWN will be executed.

How can an algorithm with low expected collision cost make a decision based on whether $\Theta(\sqrt{C}\log(w_{\text{cur}}))$ collisions occur? Recall that in this case, we are deciding between $w_{\text{cur}} \in$ ROCK-BOTTOM and $w_{\text{cur}} \in$ GOOD. We will only witness this number of collisions when w_{cur} is in the ROCK-BOTTOM range, where $\mathcal{C} \leq n$ and so we can tolerate this cost. Otherwise, as described above, $w_{\text{cur}} \in$ GOOD and the expected number of collisions is only $\tilde{O}(1/\sqrt{C})$ (see the discussion preceding Lemma 11 in our appendix).

The remaining portion of DIAGNOSIS starts with the else-statement on Line 25. This line is executed only if the conditional on Line 17 is not met; that is, we have very few successes. This can occur only if $w_{\text{cur}} \in$ ROCK-BOTTOM or $w_{\text{cur}} \in$ HIGH. The former case is (again) diagnosed be checking whether there are many collisions (Line 26) and, if so, the window is doubled; otherwise, we are in the latter case and the window is halved. Again, we only have many collisions if w_{cur} is in the ROCK-BOTTOM or lower portion of the LOW ranges, where $\mathcal{C} \leq n$ and so we can tolerate this cost.

What about the uncertain ranges? In UNCERTAIN-LOW, a sample will contain enough successes to satisfy Line 17. However, it is unclear whether the number of successes will be "large" (if w_{cur} is in the lower portion of UNCERTAIN-LOW) or "moderate" (if w_{cur} is in the upper portion of UNCERTAIN-LOW). In the former case, we fail Line 18, while in the latter case, we satisfy Line 18. Despite this uncertainty, observe that we are nonetheless guaranteed to either double the window or execute RUNDOWN. Either of these outcomes count as progress, since either w_{cur} moves closer to GOOD by doubling, or RUNDOWN is executed on a window of size $\Theta(n\sqrt{\mathcal{C}})$ (in contrast, halving w_{cur} would be counterproductive).

The intuition behind UNCERTAIN-HIGH is similar. If $w_{\text{cur}} \in$ HIGH, then we can show that the number of successes fails Line 17 (see Lemma 13) and that, ultimately, we halve w_{cur}. However, between GOOD and HIGH, this cannot be shown w.h.p.; it may hold if w_{cur} is close to the lower end of HIGH, but not hold if w_{cur} is close to the upper end of GOOD. So, instead, we argue that we either halve w_{cur} or execute RUNDOWN. Either of these actions counts as progress, since either w_{cur} moves closer to GOOD, or RUNDOWN is executed on a window of size $\Theta(n\sqrt{\mathcal{C}})$.

Ultimately, we are able to show the following key lemma (in Section A.3 of our appendix):

Lemma 19. *The executions of* COLLECT-SAMPLE *and* DIAGNOSIS *guarantee that w.h.p.: (i)* RUNDOWN *is executed within $O(\sqrt{\mathcal{C}}\log^2(n))$ slots, and (ii) the total expected collision cost until that point is $O(n\sqrt{\mathcal{C}}\log^2(n))$.*

All Remaining Active Packets Succeed. The final subroutine, **RUNDOWN**, is executed once $w_{\text{cur}} = \Theta(n\sqrt{\mathcal{C}})$, and it allows all active packets to succeed. In each slot of w_{cur}, every packet sends with probability $2/w_{\text{cur}}$. Otherwise, the current window size is halved and the remaining active packets repeat this process. This continues until the window size reaches $\Theta(\sqrt{\mathcal{C}}\log(w_0)) = \Theta(\sqrt{\mathcal{C}}\log(n))$, where the asymptotic equality holds by recalling that $\mathcal{C} = \texttt{poly}(n)$. Once this smallest window in the run is reached, $O(\log n)$ active packets remain. To finish these packets, CAB performs an additional $\Theta(\ln(n\sqrt{\mathcal{C}})) = \Theta(\ln(n))$ windows of size $\Theta(n\sqrt{\mathcal{C}})$, where any remaining active packet sends in each slot with probability $\Theta(1/n\sqrt{\mathcal{C}})$.

We prove the following lemmas (in Section A.4 of our appendix):

Lemma 21. *When* RUNDOWN *is executed, w.h.p. all packets succeed within $O(n\sqrt{\mathcal{C}}\ln(n))$ slots.*

Lemma 22. *W.h.p. the expected collision cost for executing* RUNDOWN *is $O(n\sqrt{\mathcal{C}}\ln(n))$.*

Finally, our upper bound in Theorem 1 follows directly from Lemmas 19, 21, and 22.

4 Technical Overview for Lower Bound

In this section, we provide an overview of our argument, which focuses on placing a lower bound on the expected collision cost. Here, we highlight the key lemmas in our argument, and our full proofs are provided in Section B of our appendix.

We consider only the set of slots, $\boldsymbol{\mathcal{S}}$, in the execution of any algorithm where at least two active packets remain, since we cannot have a collision with a single active packet. While we do not always make this explicit, but going forward, any slot t is assumed to implicitly belong to $\mathcal{S}$.

Let $\boldsymbol{p_i(t)}$ denote the probability that packet i sends in slot t. Note that, if a packet has terminated, its sending probability can be viewed as 0. For any fixed slot t, the ***contention*** in slot t is $\boldsymbol{\texttt{Con}(t)} = \sum_{i=1}^{n} p_i(t)$; that is, the sum of the sending probabilities in that slot.

When Contention is High. We start by showing that any algorithm that has even a single slot t with $\texttt{Con}(t) > 2$ must have $\Omega(\mathcal{C})$ expected collision cost, which is one portion of our lower bound. This is done by deriving an expression for the probability of a collision in any fixed slot t as a function of $\texttt{Con}(t)$, which is useful when $\texttt{Con}(t)$ is "high" (see Lemmas 23, 24, and 25 in Section B of our appendix).

When Contention is Low. What about an algorithm where all slots of the execution have "low" contention (i.e., $\texttt{Con}(t) \leq 2$)? Our previous expression is hard to work with in this case. So, we derive a different expression for the probability of a collision in a fixed slot t, which can be useful for small values of $\texttt{Con}(t)$:

Lemma 26. *Fix any slot t and let $\texttt{Con}(t) \leq 2$. The probability of a collision in t is at least $(\frac{1}{110})\left(\texttt{Con}(t)^2 - \sum_i p_i(t)^2\right)$.*

However, this expression requires some additional work to be deployed in our argument, as we now describe.

For any fixed slot $t \in \mathcal{S}$, let $\boldsymbol{p_{\max}(t)}$ be the maximum sending probability of any packet in slot t. Similarly, let $\boldsymbol{p_{\text{sec}}(t)} \leq p_{\max}(t)$ be the next-largest sending probability of any packet in slot t; note that $p_{\text{sec}}(t) = p_{\max}(t)$, if more than one packet sends with probability $p_{\max}(t)$.

Define $\boldsymbol{\Delta(t)} = p_{\text{sec}}(t)/p_{\max}(t)$. Our analysis ignores any slot t where $\texttt{Con}(t) = 0$; that is, any slot where every packet has a sending probability of 0. We give such slots to any algorithm for "free"; that is, we do not include them in the cost of the execution. Thus, $\Delta(t)$ is always well-defined, since $p_{\max}(t) > 0$.

Why do we need $\Delta(t)$? It captures a sense of "balance". For our purposes, the situation is most "unbalanced" when exactly one packet has non-zero sending probability, while all other packets have zero sending probability; that is, when the total value of $\texttt{Con}(t) > 0$ is due to a single packet. Clearly, in such slots, there can be no collision and, correspondingly, $\Delta(t) = 0$. In our argument, $\Delta(t)$ plays a key role in establishing the following:

Lemma 27. *For any fixed slot t, $\texttt{Con}(t)^2 - \sum_i p_i(t)^2 \geq \Delta(t)\,\texttt{Con}(t)^2/2$.*

A natural extension of $\Delta(t)$ is $\boldsymbol{\Delta_{\min}}$, which is the minimum $\Delta(t)$ over all slots $t \in \mathcal{S}$. As we show momentarily, our lower bound is parameterized by $\Delta_{\min}$. The last component of our argument addresses the sum of the contention over all slots in $\mathcal{S}$:

Lemma 30. *Any algorithm that guarantees w.h.p. that n packets succeed must w.h.p. have $\sum_{t\in\mathcal{S}} \texttt{Con}(t) = \Omega(n)$.*

We can now establish a lower bound for this low-contention case:

Lemma 31. *Consider any algorithm $\mathcal{A}$ whose contention in any slot is at most 2 and guarantees w.h.p. a makespan of $\tilde{O}(n\sqrt{\mathcal{C}})$. W.h.p. the expected collision cost for $\mathcal{A}$ is $\tilde{\Omega}(\Delta_{\min} n\sqrt{\mathcal{C}})$.*

Proof. Let X be a random variable that is $|\mathcal{S}|$ under the execution of $\mathcal{A}$. Note that, since w.h.p. the makespan is $\tilde{O}(n\sqrt{\mathcal{C}})$, it must be the case that w.h.p. $X = \tilde{O}(n\sqrt{\mathcal{C}})$.

By Lemma 30, we have:

$$\sum_{t=1}^{X} \texttt{Con}(t) \geq cn. \tag{1}$$

for some constant $c > 0$. Let $Y_t = 1$ if slot $t \in \mathcal{S}$ has a collision; otherwise, $Y_t = 0$. By Lemmas 26 and 27:

$$\begin{aligned} Pr(Y_t = 1) &\geq \Delta(t) \cdot \texttt{Con}(t)^2/220 \\ &\geq \Delta_{\min} \cdot \texttt{Con}(t)^2/220 \end{aligned}$$

where the second line follows from the definition of $\Delta_{\min}$. The expected collision cost is:

$$\begin{aligned} \sum_{t=1}^{X} P(Y_t = 1) \cdot \mathcal{C} &\geq \sum_{t=1}^{X} \frac{\Delta_{\min}\texttt{Con}(t)^2 \cdot \mathcal{C}}{220} \\ &= \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \sum_{t=1}^{X} \texttt{Con}(t)^2. \end{aligned} \tag{2}$$

By Jensen's inequality for convex functions, we have:

$$\frac{\sum_{t=1}^{X} \texttt{Con}(t)^2}{X} \geq \left(\frac{\sum_{t=1}^{X} \texttt{Con}(t)}{X}\right)^2 \tag{3}$$

Finally, the expected cost is at least:

$$\begin{aligned} \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \sum_{t=1}^{X} \texttt{Con}(t)^2 &\geq \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \frac{\left(\sum_{t=1}^{X} \texttt{Con}(t)\right)^2}{X} \quad \text{by Equations 8 and 9} \\ &\geq \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \left(\frac{c^2 n^2}{X}\right) \quad \text{by Equation 7} \\ &= \tilde{\Omega}\left(\Delta_{\min}\, n\sqrt{\mathcal{C}}\right) \end{aligned}$$

where the second line follows by Equation 7, which was defined with regard to $\mathcal{S}$, and so can be compared to Equation 8. The last line follows since w.h.p. $X = \tilde{O}(n\sqrt{\mathcal{C}})$. □

Given our analysis of the high- and low-contention cases, we have the following lower bound:

Theorem 4. *Consider any algorithm that w.h.p. guarantees $\tilde{O}(n\sqrt{\mathcal{C}})$ makespan. Then, w.h.p., the expected collision cost for $\mathcal{A}$ is $\tilde{\Omega}(\min\{\mathcal{C}, \Delta_{\min} n\sqrt{\mathcal{C}}\})$.*

What algorithms does our lower bound say something interesting about? We can start with our statement in Theorem 2. For a well-known lower bound with the standard cost model, Willard [35] defines and argues about ***fair*** algorithms. These are algorithms where, in a fixed slot, every active packet sends with the same probability (and the probability may change from slot to slot). Fair algorithms have $\Delta_{\min} = 1$, and for a sufficiently large collision cost—specifically, $\mathcal{C} = \Omega(n^2)$—the lower bound becomes $\tilde{\Omega}(n\sqrt{\mathcal{C}})$. Given that CAB is fair and w.h.p. guarantees $\tilde{O}(n\sqrt{\mathcal{C}})$ makespan, we thus have a lower bound that is asymptotically tight to a $\texttt{poly}(\log n)$-factor with our upper bound.

Our lower bound also applies in a similar way to a generalized notion of fairness, where the sending probabilities of any two packets are within some factor $\delta > 0$. For example, if $\delta = \Theta(1)$, then $\Delta_{\min} = \Theta(1)$. Indeed, we only need such δ-fairness between the two packets with the largest probabilities for our lower bound to apply, although our bound weakens as δ grows larger.

Another class of algorithms that our lower bound applies to is ***multiplicative weights update algorithms***, where in each slot every packet updates its sending probability by a multiplicative factor based on channel feedback in the slot. Many of these algorithms are fair (such as [18, 11, 34, 33, 31, 7, 13]), but not all are (such as [11]). Our lower bound applies in a non-trivial way to all such algorithms; that is, $\Delta_{\min} > 0$ given the update rules.

5 Conclusion and Future Work

We considered a model for the problem contention resolution where each collision has cost $\mathcal{C}$. Our algorithm, COLLISION-AVERSION BACKOFF, addresses the static case and guarantees w.h.p. that all packets succeed with makespan and expected collision cost that is $\tilde{O}(n\sqrt{\mathcal{C}})$.

There are several directions for future work that are potentially interesting. First, we would like to extend this cost model to the dynamic setting (where packets may arrive over time) and design solutions. Many algorithms for the dynamic case break the packet arrivals into (mostly) disjoint batches; however, this requires periodic broadcasting that can cause collisions.

Second, for the lower bound, we believe (with some more work) that we can remove the $\texttt{poly}(\log n)$ factor. However, we would like to derive a more general lower bound. A significant challenge appears to be addressing algorithms that set up slots where exactly one packet has non-zero sending probability, while all others have zero sending probability. For example, an elected leader might "schedule" each packet their own exclusive slot in which to send (with probability 1). Since there can be no collisions after such a schedule is implemented, it seems a lower bound argument must demonstrate that establishing a schedule is costly.

Third, what if collisions are not fully dictated by the actions of packets? Some wireless settings are inherently "noisy", due to weather conditions, co-located networks, or faulty devices. Is there a sensible model for such settings and, if so, can we reduce the cost from collisions?

Acknowledgements. We are grateful to the anonymous reviewers for their feedback on our manuscript. This work is supported by NSF award CCF-2144410.

References

1. Abramson, N.: The ALOHA system: Another alternative for computer communications. In: Proceedings of the November 17-19, 1970, fall joint computer conference. pp. 281–285 (1970)
2. Aldawsari, B.A., Chlebus, B.S., Kowalski, D.R.: Broadcasting on adversarial multiple access channels. In: Proceedings of the IEEE 18th International Symposium on Network Computing and Applications (NCA). pp. 1–4 (2019)
3. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Deterministic broadcast on multiple access channels. In: Proceedings IEEE INFOCOM. pp. 1–5. IEEE (2010)
4. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Medium access control for adversarial channels with jamming. In: Proceedings of the International Colloquium on Structural Information and Communication Complexity. pp. 89–100. Springer (2011)
5. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Packet latency of deterministic broadcasting in adversarial multiple access channels. Journal of Computer and System Sciences **99**, 27–52 (2019)

6. Anderton, W.C., Chakraborty, T., Young, M.: Windowed backoff algorithms for WiFi: Theory and performance under batched arrivals. Distributed Computing **34**, 367–393 (2021)
7. Awerbuch, B., Richa, A., Scheideler, C.: A jamming-resistant MAC protocol for single-hop wireless networks. In: Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC). pp. 45–54 (2008)
8. Ben-David, N., Blelloch, G.E.: Analyzing contention and backoff in asynchronous shared memory. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 53–62 (2017)
9. Bender, M.A., Farach-Colton, M., He, S., Kuszmaul, B.C., Leiserson, C.E.: Adversarial contention resolution for simple channels. In: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). pp. 325–332 (2005)
10. Bender, M.A., Fineman, J.T., Gilbert, S.: Contention resolution with heterogeneous job sizes. In: Proceedings of the 14th Conference on Annual European Symposium (ESA). pp. 112–123 (2006)
11. Bender, M.A., Fineman, J.T., Gilbert, S., Kuszmaul, J., Young, M.: Fully energy-efficient randomized backoff: Slow feedback loops yield fast contention resolution. In: Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC). p. 231–242 (2024). https://doi.org/10.1145/3662158.3662807
12. Bender, M.A., Fineman, J.T., Gilbert, S., Young, M.: How to Scale Exponential Backoff: Constant Throughput, Polylog Access Attempts, and Robustness. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 636–654. SODA '16 (2016)
13. Bender, M.A., Gilbert, S., Kuhn, F., Kuszmaul, J., Médard, M.: Contention resolution for coded radio networks. In: Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). pp. 119–130. ACM (2022)
14. Bender, M.A., Kopelowitz, T., Kuszmaul, W., Pettie, S.: Contention resolution without collision detection. In: Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC). pp. 105–118 (2020)
15. Bender, M.A., Kopelowitz, T., Pettie, S., Young, M.: Contention resolution with log-logstar channel accesses. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing. pp. 499–508. STOC 2016 (2016)
16. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE Journal on selected areas in communications **18**(3), 535–547 (2000)
17. Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., Williamson, D.P.: Adversarial queuing theory. Journal of the ACM (JACM) **48**(1), 13–38 (2001)
18. Chang, Y., Jin, W., Pettie, S.: Simple contention resolution via multiplicative weight updates. In: Proceedings of the Second Symposium on Simplicity in Algorithms (SOSA). pp. 16:1–16:16 (2019)
19. Chen, H., Jiang, Y., Zheng, C.: Tight trade-off in contention resolution without collision detection. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 139–149 (2021)
20. Chlebus, B.S.: Randomized communication in radio networks. Handbook of Randomized Computing **1**, 401–456 (2001)
21. Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Stability of the multiple-access channel under maximum broadcast loads. In: Proceedings of the Symposium on Self-Stabilizing Systems (SSS). pp. 124–138. Springer (2007)
22. Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Maximum throughput of multiple access channels in adversarial environments. Distributed Computing **22**(2), 93–116 (2009)
23. De Marco, G., Stachowiak, G.: Asynchronous shared channel. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 391–400. PODC '17 (2017)
24. Dwork, C., Herlihy, M., Waarts, O.: Contention in shared memory algorithms. Journal of the ACM (JACM) **44**(6), 779–805 (1997)
25. Fineman, J.T., Newport, C., Wang, T.: Contention resolution on multiple channels with collision detection. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing. pp. 175–184. PODC '16 (2016)
26. Geréb-Graus, M., Tsantilas, T.: Efficient optical communication in parallel computers. In: Proceedings of the 4th Annual ACM Symposium on Parallel algorithms and Architectures. pp. 41–48 (1992)
27. Goemans, M.: Chernoff bounds, and some applications. https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf (viewed May 2024) (Feb 2015)
28. Greenberg, R.I., Leiserson, C.E.: Randomized routing on fat-trees. In: Proceedings of the 26th Annual Symposium on the Foundations of Computer Science (FOCS). pp. 241–249 (1985)
29. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-efficient size approximation of radio networks with no collision detection. In: Proceedings of the 8th Annual International Conference (COCOON). pp. 279–289 (2002)
30. Metcalfe, R.M., Boggs, D.R.: Ethernet: Distributed packet switching for local computer networks. CACM **19**(7), 395–404 (July 1976)

31. Ogierman, A., Richa, A., Scheideler, C., Schmid, S., Zhang, J.: Competitive MAC under adversarial SINR. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM). pp. 2751–2759 (2014)
32. Ramaiyan, V., Vaishakh, J.: An information theoretic point of view to contention resolution. In: 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS). pp. 1–8. IEEE (2014)
33. Richa, A., Scheideler, C., Schmid, S., Zhang, J.: An Efficient and Fair MAC Protocol Robust to Reactive Interference. IEEE/ACM Transactions on Networking **21**(1), 760–771 (2013)
34. Richa, A., Scheideler, C., Schmid, S., Zhang, J.: Competitive Throughput in Multi-Hop Wireless Networks Despite Adaptive Jamming. Distributed Computing **26**(3), 159–171 (2013)
35. Willard, D.E.: Log-logarithmic selection resolution protocols in a multiple access channel. SIAM J. Comput. **15**(2), 468–477 (May 1986)
36. Zhang, Z.: Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. IEEE Communications Surveys & Tutorials **8**(1), 24–37 (2006)

Appendix

A Analysis for Our Upper Bound

In this part of the appendix, we provide our full proofs for our upper bound, which culminate in Theorem 1. We begin in Section A.1 by addressing mathematical tools that are useful for our analysis. In Section A.2, we argue that w.h.p. DIAGNOSIS correctly determines the current range and takes the correct action. We follow up in Section A.3 with a cost analysis for the executions of COLLECT-SAMPLE. In Section A.4, we analyze RUNDOWN, showing that w.h.p. all packets succeed, along with a bound on the expected cost. Finally, in Section A.5, we put all of these pieces together in order to establish the correctness and bounds on the expected cost for COLLISION-AVERSION BACKOFF

For ease of presentation, our analysis assumes pessimistically that no packets succeed until they start executing RUNDOWN. However, our results hold if we allow packets to succeed (and then terminate) earlier in the execution COLLECT-SAMPLE, but this number is negligible (being $\tilde{O}(\sqrt{n})$).

A.1 Preliminaries

Throughout, we assume that n is sufficiently large. We make use of the following well-known facts.

Fact 1: *The following inequalities hold.*

(a) For any x, $1 - x \leq e^{-x}$,
(b) For any $0 \leq x < 1$, $1 - x \geq e^{-x/(1-x)}$,
(c) For any $0 \leq x \leq 1/2$, $1 - x \geq e^{-2x}$.

Note that Fact 1(c) follows directly from Fact 1(b); however, the former is sometimes easier to employ, and we state it explicitly to avoid any confusion. We also make use of the following standard Chernoff bounds, stated below.

Theorem 3. *([27]) Let* $X = \sum_i X_i$, *where* X_i *are Bernoulli indicator random variables with probability* p. *The following holds:*

$$Pr\left(X \geq (1+\delta)E[X]\right) \leq \exp\left(-\frac{\delta^2 E[X]}{2+\delta}\right) \quad \textit{for any } \delta > 0$$

and

$$Pr\left(X \leq (1-\delta)E[X]\right) \leq \exp\left(-\frac{\delta^2 E[X]}{2+\delta}\right) \quad \textit{for any } 0 < \delta < 1. \square$$

Throughout, we use with high probability to mean with probability $1 - O(1/n^{\kappa+1})$, where n^{κ} is an upper bound on $\mathcal{C}$ (recall Section 1.1).

In our first lemma below, we prove an upper bound on the probability of a collision in any slot as a function of the number of active packets. This result is a useful tool in our later arguments.

Lemma 1. *Consider any slot* t, *where there exist* m *active packets, each sending with the same probability* p, *where* $p < 1/m$. *The probability of a collision in slot* t *is at most* $\frac{2m^2p^2}{(1-mp)}$.

Proof. Fix a time slot t. Let p_i denote the probability that packet i sends in slot t and number of packets at the beginning is m. The probability of collision in slot t is at most:

$$\begin{aligned}
&1 - Pr(\text{success in slot } t) - Pr(\text{empty slot in slot } t) \\
&= 1 - \binom{m}{1} p^1 (1-p)^{(m-1)} - \binom{m}{0} p^0 (1-p)^m \\
&\leq 1 - mp(1-p)^m - (1-p)^m \\
&= 1 - (1-p)^m (mp+1) \\
&\leq 1 - e^{-\frac{mp}{1-p}} (mp+1) \quad \text{by Fact 1(b)} \\
&\leq 1 - \left(1 - \frac{mp}{1-p}\right)(mp+1) \\
&\leq 1 - \left(1 - \frac{mp}{1-mp}\right)(mp+1) \quad \text{for } mp < 1 \\
&= 1 - \left(mp - \frac{m^2p^2}{1-mp} + 1 - \frac{mp}{1-mp}\right) \\
&= \frac{m^2p^2}{1-mp} - mp + \frac{mp}{1-mp} \\
&= \frac{m^2p^2 - mp + m^2p^2 + mp}{1-mp} \\
&= \frac{m^2p^2 - mp + m^2p^2 + mp}{1-mp} \\
&= \frac{2m^2p^2}{(1-mp)}
\end{aligned}$$

which proves the claim. □

A.2 Correctness Analysis for DIAGNOSIS

In order to argue correctness for DIAGNOSIS, we give a case analysis for each of the ROCK-BOTTOM, LOW, GOOD, and HIGH ranges. In each case, we provide the necessary bounds on successes and collision per sample, which allows us to demonstrate that DIAGNOSIS correctly diagnoses the current range and performs the correct action. We start with Lemmas 2, 3, 4, and 5, which establish lower and upper bounds on the number of successes in the ranges of LOW, GOOD, and HIGH.

Lemma 2. *For any window size $w_{cur} = xn\sqrt{C}$ where $x > 0$ is a constant, the expected number of successes in a sample of size $d\sqrt{C}\ln(w_{cur})$ at most $\frac{d\ln(w_{cur})}{x} e^{-1/(2x\sqrt{C})}$.*

Proof. Let X_i be an independent random variable where $X_i = 1$ when i-th slot in w contains a success, and $X_i = 0$ otherwise.

$$\begin{aligned}
Pr(X_i = 1) &= \binom{n}{1} \left(\frac{1}{xn\sqrt{C}}\right) \left(1 - \frac{1}{xn\sqrt{C}}\right)^{n-1} \\
&\leq \frac{1}{x\sqrt{C}} e^{-(n-1)/(xn\sqrt{C})} \quad \text{by Fact 1(a)} \\
&\leq \frac{1}{x\sqrt{C}} e^{-(n/2)/(xn\sqrt{C})} \quad \text{since } n \geq 2 \\
&= \frac{1}{x\sqrt{C}} e^{-1/(2x\sqrt{C})}
\end{aligned}$$

Let $X = \sum_{k=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_k$, then the expected number of success:

$$\begin{aligned} E[X] &\leq E\left[\sum_{k=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_k\right] \\ &= \sum_{k=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} E[X_k] \quad \text{by linearity of expectation} \\ &= \frac{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})}{x\sqrt{\mathcal{C}}} e^{-1/(2x\sqrt{\mathcal{C}})} \\ &= \frac{d\ln(w_{\text{cur}})}{x} e^{-1/(2x\sqrt{\mathcal{C}})} \end{aligned}$$

which proves the claim. □

Lemma 3. *For any window size $w_{cur} = xn\sqrt{\mathcal{C}}$ where $x > 0$ is a constant, the expected number of successes in a sample is at least $\frac{d\ln(w_{cur})}{x} e^{-1/(x\sqrt{\mathcal{C}})}$.*

Proof. Let X_i be an independent random variable where $X_i = 1$ when i-th slot in w contains a success, and $X_i = 0$ otherwise.

$$\begin{aligned} Pr(X_i = 1) &= \binom{n}{1}\left(\frac{1}{xn\sqrt{\mathcal{C}}}\right)\left(1 - \frac{1}{xn\sqrt{\mathcal{C}}}\right)^{n-1} \\ &\geq \frac{1}{x\sqrt{\mathcal{C}}} e^{-2(n-1)/(xn\sqrt{\mathcal{C}})} \quad \text{by Fact 1(c)} \\ &\geq \frac{1}{x\sqrt{\mathcal{C}}} e^{-1/(x\sqrt{\mathcal{C}})}. \end{aligned}$$

Let X_k be an independent random variable where $X_k = 1$ when k-th slot in the sample contains a success, and $X_k = 0$ otherwise. The expected number of successes in the sample is:

$$\begin{aligned} E[X] &\geq E\left[\sum_{k=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_k\right] \\ &= \sum_{k=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} E[X_k] \quad \text{by linearity of expectation} \\ &\geq \frac{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})}{x\sqrt{\mathcal{C}}} e^{-1/(x\sqrt{\mathcal{C}})} \\ &= \frac{d\ln(w_{\text{cur}})}{x} e^{-1/(x\sqrt{\mathcal{C}})} \end{aligned}$$

which proves the claim. □

Lemma 4. *If $w_{cur} \geq an\sqrt{\mathcal{C}}$, where $a \geq 1$ is a constant, then with probability at least $1 - 1/n^{d'}$ the number of successes in a sample is at most $(2d/a)\ln(w_{cur})$, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. Let $x = a$ in Lemma 2, which implies that the expected number of successes in the sample is at most:

$$\frac{d\ln(w_{\text{cur}})}{a} e^{-1/(2a\sqrt{\mathcal{C}})} \leq \frac{d\ln(w_{\text{cur}})}{a}.$$

Let $X_i = 1$ if the i-th slot of the sample contains a success; otherwise, let $X_i = 0$. Let $X = \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i$, and note by the above that $E[X] \leq \frac{d\ln(w_{\text{cur}})}{a}$. By Theorem 3, with $\delta = 1$ below, we have:

$$\begin{aligned}
Pr\left(X \geq (1+\delta)\frac{d\ln(w_{\text{cur}})}{a}\right) &\leq e^{-\frac{(1)^2 d\ln(w_{\text{cur}})}{(1+2)a}} \\
&= e^{-\frac{d\ln(w_{\text{cur}})}{3a}} \\
&= w_{\text{cur}}^{-\frac{d}{3a}} \\
&\leq n^{-\frac{d}{3a}} \quad \text{since } w_{\text{cur}} \geq n \\
&= n^{-d'}
\end{aligned}$$

where d' can be an arbitrarily large constant, so long as d is sufficiently large. □

Our next lemma provides a useful lower bound on the number of successes.

Lemma 5. *If $n \leq w \leq bn\sqrt{C}$ for a constant $b \geq 10$, then with probability at least $1 - 1/n^{d'}$ the number of successes exceeds $(d/2be)\ln(w_{cur})$, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. Let $x = b$ in Lemma 3, which implies that the expected number of successes in the sample is at least:

$$\frac{d\ln(w_{\text{cur}})}{b}e^{-1/(b\sqrt{C})} > \frac{d\ln(w_{\text{cur}})}{e\,b}$$

for any $b \geq 10$. Let $X_i = 1$ if the i-th slot of the sample contains a success; otherwise, let $X_i = 0$. Let $X = \sum_{i=1}^{d\sqrt{C}\ln(w_{\text{cur}})} X_i$, and note by the above that $E[X] \geq \frac{d\ln(w_{\text{cur}})}{be}$. By Theorem 3, with $\delta = 1/2$, we have:

$$\begin{aligned}
Pr\left(X \leq (1-\delta)\frac{d\ln(w_{\text{cur}})}{be}\right) &\leq e^{-\frac{(1/2)^2 d\ln(w_{\text{cur}})}{(2+1/2)eb}} \\
&= e^{-\frac{d\ln(w_{\text{cur}})}{10eb}} \\
&= w_{\text{cur}}^{-\frac{d}{10eb}} \\
&\leq n^{-\frac{d}{10eb}} \\
&= \frac{1}{n^{d'}}
\end{aligned}$$

where d' can be an arbitrarily large constant, so long as d is sufficiently large. □

The next two lemmas pertain to the range of ROCK-BOTTOM. Lemma 6 shows that, when the window size is $O(n/\log n)$, the entire sample will consist of collisions. Unlike many of our other arguments, we cannot employ a Chernoff to establish this fact. Lemma 7 handles the remainder of the ROCK-BOTTOM range, showing that much of the sample will consist of collisions.

Lemma 6. *Suppose that $1 \leq w_{cur} \leq \frac{n}{c\ln n}$ and n is sufficiently large and $c \geq 1$ is a constant. Then, with probability at least $1 - 1/n^{d'}$, the entire sample consists of collisions, where d' depends on sufficiently large c.*

Proof. For any fixed slot, the probability of a collision is at least 1 minus the probability of a success, minus the probability of that the slot is empty. In other words, the probability of a collision is at least:

$$\begin{aligned}
&1 - \left(1 - \frac{c\ln n}{n}\right)^n - \binom{n}{1}\left(\frac{c\ln n}{n}\right)\left(1 - \frac{c\ln n}{n}\right)^{n-1} \\
&\geq 1 - \frac{1}{e^{c\ln n}} - \left(\frac{c\ln n}{e^{c(n-1)\ln(n)/n}}\right) \quad \text{by Fact 1(a)} \\
&\geq 1 - \frac{1}{e^{c\ln n}} - \left(\frac{c\ln n}{e^{c\ln(n)/2}}\right) \quad \text{since } \tfrac{n-1}{n} \geq \tfrac{1}{2} \text{ for } n \geq 2 \\
&= 1 - \frac{1}{n^c} - \frac{c\ln n}{n^{c/2}} \\
&\geq 1 - \frac{1}{n^{d'}} \quad \text{for } n \text{ sufficiently large}
\end{aligned}$$

which yields the claim. □

Lemma 7. *Suppose $\frac{n}{c \ln n} < w_{cur} \leq n$, where $c \geq 1$ is any constant. Then, with probability at least $1 - 1/n^{d'}$, the sample contains at least $d\sqrt{\mathcal{C}} \ln(w_{cur})/(8e^2)$ slots consists of collisions, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. Let the indicator random variable $X_i = 1$ if the i-th slot of the $d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})$ slots contains a collision; otherwise, let $X_i = 0$. The probability of a collision is lower bounded as follows.

$$\begin{aligned}
Pr(X_i = 1) &\geq \binom{n}{2} \left(\frac{1}{w_{\text{cur}}}\right)^2 \left(1 - \frac{1}{w_{\text{cur}}}\right)^{n-2} \\
&\geq \binom{n}{2} \left(\frac{1}{w_{\text{cur}}}\right)^2 \left(1 - \frac{1}{w_{\text{cur}}}\right)^{n} \\
&\geq \left(\frac{n(n-1)}{2w_{\text{cur}}^2}\right) e^{-2n/w_{\text{cur}}} \quad \text{by Fact 1(c)} \\
&\geq \frac{(n-1)}{2ne^2} \quad \text{minimized when } w_{\text{cur}} = n \\
&\geq \frac{1}{4e^2} \quad \text{since } \tfrac{n-1}{n} \geq \tfrac{1}{2} \text{ for } n \geq 2.
\end{aligned}$$

Let $X = \sum_{i=1}^{d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})} X_i$. The expected number of collisions in the sample of $d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})$ slots is:

$$\begin{aligned}
E[X] &\geq E\left[\sum_{i=1}^{d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})} X_i\right] \\
&= \frac{d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})}{4e^2} \quad \text{by linearity of expectation.}
\end{aligned}$$

We then employ Theorem 3, with $\delta = 1/2$, to obtain:

$$\begin{aligned}
Pr\left(X \leq (1-\delta)\frac{d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})}{4e^2}\right) &\leq e^{-\frac{(1/2)^2 d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})}{(5/2)4e^2}} \\
&= e^{-\frac{d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})}{40e^2}} \\
&= w_{\text{cur}}^{-\frac{d\sqrt{\mathcal{C}}}{40e^2}} \\
&\leq (n/c \log n)^{-\frac{d\sqrt{\mathcal{C}}}{40e^2}} \\
&\leq n^{-d'}
\end{aligned}$$

where d' can be an arbitrarily large constant, so long as d is sufficiently large. □

Lemma 8. *Under* DIAGNOSIS*, all packets witness the same number of empty slots, successes, and collisions.*

Proof. In each slot, a packet is either sending or listening. First, consider the case where at least one packet sends. Each such sending packet knows whether it succeeded (i.e., a success) or it failed (i.e., a collision). Each non-sending packet listens and hears the success (i.e., a success) or the failure (i.e., a collision). Second, if no packet sends, then every packet is listening and hears the same thing: silence. Therefore, in all cases, each packet witnesses the same outcome of the slot. □

Lemma 8 guarantees that all packets are always in the same window (or have the same sending probability) when executing DIAGNOSIS, and they always take the same action in an execution of DIAGNOSIS. The following lemmas establish that DIAGNOSIS takes the *correct* action in each execution.

Lemma 9. *With probability at least $1 - O(1/n^{d'})$, if $w_{cur} \in$ ROCK-BOTTOM, then all packets double their window, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. We do a case analysis. For the first case, consider that $w_{\text{cur}} \in [1, n/(c \ln n))$ for some sufficient large constant $c \geq 1$. By Lemma 6, with probability at least $1 - n^{-d'}$, all slots in the sample contain a collision. Therefore, w.h.p., the if-statement on Line 17 is not entered and, instead, the matching else-statement on Line 25 is entered. Since all sample slots are collisions, the if-statement on Line 26 is entered, and the window doubles.

For the second case, consider $w_{\text{cur}} \in [n/(c \ln n), n)$. Either we (a) enter the first if-statement on Line 17, or we (b) enter the matching else-statement on Line 25.

- **Subcase (a).** If the second if-statement on Line 18 is entered, then we note by Lemma 7, w.h.p., that we have at least $d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})/(8e^2)$ collisions, which means we double the window, as desired. Otherwise, we execute Line 23, and the window doubles.
- **Subcase (b).** By Lemma 7 w.h.p. we have at least $d\sqrt{\mathcal{C}} \ln(w_{\text{cur}})/(8e^2)$ collisions, which means we enter the if-statement on Line 26 and double the window.

In both cases, the packet doubles its window. □

Lemma 10. *With probability $1 - O(1/n^{d'})$, if $w_{cur} \in$ LOW, then all packets double their window, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. By Lemma 5 with $b = 10$, for w such that $n \leq w < 10n\sqrt{\mathcal{C}}$, with probability at least $1 - n^{-d'}$ the number of successes exceeds $\frac{d}{20e} \ln(w_{\text{cur}})$. Therefore, the first if-statement on Line 17 is entered, but the next if-statement on Line 18 is not. Thus, Line 23 will be executed and so the window will double, as claimed. □

The following lemma places an upper bound on the number of collisions when the window is at least $200n\sqrt{\mathcal{C}}$. This allows us to argue that when we are in the GOOD and HIGH ranges, we avoid executing the lines in DIAGNOSIS that would double the window.

We note that the next lemma gives a very loose upper bound on the number of collisions. In fact, if we expected even a single collision in this range, our claimed upper bound would not hold. The careful reader will note that (in our proof) the expected number of collisions is $\tilde{O}(1/\sqrt{\mathcal{C}})$; however, a loose upper bound that holds with high probability is sufficient to construct the rules for DIAGNOSIS.

Lemma 11. *With probability at least $1 - 1/n^{d'}$, if $w_{cur} \geq 200n\sqrt{\mathcal{C}}$, then the number of collisions in the sample is at most $(d/90)\ln(w_{cur})$, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. By Lemma 1, the probability of a collision in a slot is at most $\frac{2n^2p^2}{(1-np)}$, where $p \leq 1/(200n\sqrt{\mathcal{C}})$. Plugging in, we obtain that the probability of collision is upper bounded by:

$$\begin{aligned}
\frac{2n^2p^2}{(1-np)} &\leq \frac{2n^2(1/(200n\sqrt{\mathcal{C}}))^2}{\left(1 - 1/(200\sqrt{\mathcal{C}})\right)} \\
&= \frac{1}{200\mathcal{C}(1 - 1/200)} \\
&\leq \frac{1}{100\mathcal{C}}
\end{aligned}$$

Let the indicator variable $X_i = 1$ if the i-th slot is a collision; otherwise, let $X_i = 0$. Let $X = \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i$. The expected number of collisions in the sample of $d\sqrt{\mathcal{C}} \ln w_{\text{cur}}$ slots is:

$$\begin{aligned}
E[X] &\leq E\left[\sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i\right] \\
&= \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} E[X_i] \quad \text{by linearity of expectation} \\
&\leq \frac{d\ln(w_{\text{cur}})}{100\sqrt{\mathcal{C}}} \\
&\leq \frac{d\ln(w_{\text{cur}})}{100}
\end{aligned}$$

Letting $\delta = 1/10$ in Theorem 3, we have:

$$\begin{aligned} Pr\left(X \geq (1+1/10)\frac{d\ln(w_{\text{cur}})}{100}\right) &\leq Pr\left(X \geq \frac{d\ln(w_{\text{cur}})}{90}\right) \\ &\leq e^{-\frac{(1/10)^2 d\ln(w_{\text{cur}})}{2+(1/10)}} \\ &\leq e^{-\frac{d\ln(w_{\text{cur}})}{210}} \\ &\leq \frac{1}{w_{\text{cur}}^{d'}} \\ &\leq \frac{1}{n^{d'}} \quad \text{since } w_{\text{cur}} \geq n \end{aligned}$$

where d' can be arbitrarily large depending only on d. □

We now argue that DIAGNOSIS takes the correct action in each of the GOOD and HIGH ranges.

Lemma 12. *With probability at least $1 - O(1/n^{d'})$, if $w_{cur} \in$ GOOD, then all packets execute* RUNDOWN*, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. Since $w_{\text{cur}} \in$ GOOD, this means that $200n\sqrt{\mathcal{C}} \leq w_{\text{cur}} \leq 10^3 n\sqrt{\mathcal{C}}$. By Lemma 5 with $b = 10^3$, the number of successes in the sample exceeds $\frac{d\ln w_{\text{cur}}}{2e\,10^3} > \frac{2d\ln(w_{\text{cur}})}{10^5}$. Therefore, the if-statement on Line 17 is entered. By Lemma 4 with $a = 200$, the number of successes is at most $\frac{2d\ln(w_{\text{cur}})}{200} = \frac{d\ln(w_{\text{cur}})}{100} \leq \frac{d\ln(w_{\text{cur}})}{20e}$; thus, the if-statement on Line 18 is entered. By Lemma 11, the number of collisions is at most $(d/90)\ln(w_{\text{cur}}) < d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})/(8e^2)$, so Line 19 is not entered, and instead Line 21 is entered, which executes RUNDOWN. □

Lemma 13. *With probability at least $1 - O(1/n^{d'})$, if $w_{cur} \in$ HIGH, then all packets halve their window, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. If $w_{\text{cur}} \in$ HIGH, then $w_{\text{cur}} > 10^5 n\sqrt{\mathcal{C}}$. By Lemma 4 with $a = 10^5$, the number of successes is at most $(2d/10^5)\ln(w_{\text{cur}})$. Thus, the else-statement on Line 25 is entered. By Lemma 11, the number of collisions is at most $(d/90)\ln(w_{\text{cur}}) < \frac{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})}{8e^2}$, and so Line 28 is entered and the window is halved. The total error associated with the lemmas invoked is $O(1/n^{d'})$. □

Finally, what about the "uncertain" intervals $[10n\sqrt{\mathcal{C}}, 200n\sqrt{\mathcal{C}})$ and $(10^3 n\sqrt{\mathcal{C}}, 10^5 n\sqrt{\mathcal{C}}]$? Our next two lemmas address this aspect by showing that the DIAGNOSIS revises the current window towards the GOOD or executes RUNDOWN directly.

Lemma 14. *With probability at least $1 - O(1/n^{d'})$, if $w_{cur} \in$ UNCERTAIN-LOW, then either the window doubles or* RUNDOWN *is executed, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. Since $w_{\text{cur}} \in$ UNCERTAIN-LOW, we have that $w_{\text{cur}} \in [10n\sqrt{\mathcal{C}}, 200n\sqrt{\mathcal{C}})$. By Lemma 5 with $b = 200$ (the part of UNCERTAIN-LOW where successes are most scarce), the number of successes in the sample still exceeds $\frac{d\ln w_{\text{cur}}}{400e}$, which exceeds $\frac{2d\ln(w_{\text{cur}})}{10^5}$, so Line 17 will be true. For the remaining lines that may be executed—Lines 18 to 24— we either double the window (due to many collisions) or execute RUNDOWN, as claimed. □

Lemma 15. *With probability at least $1 - O(1/n^{d'})$, if $w_{cur} \in$ UNCERTAIN-HIGH, then either the window halves or* RUNDOWN *is executed, where d' is an arbitrarily large constant depending on sufficiently large d.*

Proof. We prove this by ruling out all the ways in which w_{cur} could (incorrectly) double.

Unlike our argument for $w_{\text{cur}} \in$ GOOD, we cannot guarantee that Line 17 will be executed, since our window might be large, say just shy of $10^5 n\sqrt{\mathcal{C}}$, (recall that UNCERTAIN-HIGH $= [10^3 n\sqrt{\mathcal{C}}, 10^5 n\sqrt{\mathcal{C}})$), and we cannot prove a sufficient number of successes. However, if we proceed to Line 25, then by Lemma 11, the number of collisions is at most $(d/90)\ln(w_{\text{cur}})$, which means there are not enough collisions to meet the if-statement condition on 26, and thus the window will not be doubled here.

Another way in which the window can (incorrectly) double is via Line 23. However, we only execute Line 23 if Line 18 fails to hold; that is, in the case that the sample contains *more* than than $\frac{d\ln(w_{\text{cur}})}{20e}$ successes. However, by

Lemma 4 with $a = 10^3$ (the part of UNCERTAIN-HIGH where successes are most likely), the sample contains at most $\frac{2d\ln(w_{\text{cur}})}{10^3} < \frac{d\ln(w_{\text{cur}})}{20e}$ successes, and so Line 18 will indeed execute.

Finally, in the case that Line 18 executes, we do not have enough collisions to satisfy Line 26, again by Lemma 11. Thus, the window cannot double via Line 20.

We have shown that, when $w_{\text{cur}} \in$ UNCERTAIN-HIGH, either the window must halve, or RUNDOWN is executed, as claimed. □

A.3 Cost Analysis for COLLECT-SAMPLE

The next lemmas establish bounds on the expected collision cost for executing COLLECT-SAMPLE. To this end, we employ the results established above, which hold with probability at least $1 - 1/n^{d'}$ for as large a constant $\boldsymbol{d'} \geq 1$ as we wish, depending only on sufficiently large d. Thus, the results in this section also hold with this high probability, and for ease of presentation, we omit this from our technical lemma statements and arguments.

Lemma 16. *If $w_{\text{cur}} \in$ ROCK-BOTTOM $\cup$ LOW, the execution of* COLLECT-SAMPLE *has $O(n\sqrt{\mathcal{C}}\ln(n))$ expected collision cost.*

Proof. We prove this using a case analysis.

Case 1: $\mathcal{C} \leq 2n$. This captures all of ROCK-BOTTOM and the left endpoint of LOW. Here, even if the entire sample consists of collisions, then the cost is at most:

$$\begin{aligned}(d\sqrt{\mathcal{C}}\ln(w_{\text{cur}}))\mathcal{C} &\leq (d\sqrt{\mathcal{C}}\ln(n\sqrt{\mathcal{C}}))2n \\ &= O(n\sqrt{\mathcal{C}}\ln(n))\end{aligned}$$

Case 2: $\mathcal{C} > 2n$. With high probability, $w_{\text{cur}} \in$ LOW only if the initial window lies in LOW. (With high probability, we cannot have started in ROCK-BOTTOM and moved up, since we start at a window of size $\mathcal{C} > n$ in this case.)

The probability of a collision is maximized when the window is smallest; that is, when the window has size $\mathcal{C}$ (which is the initial window size), since w.h.p. the window only increases (given that we are in LOW). By Lemma 1, the probability of a collision is at most:

$$\begin{aligned}\frac{2n^2p^2}{1-np} &\leq \frac{2n^2(1/\mathcal{C})^2}{1-n(1/\mathcal{C})} \\ &\leq 4n^2/\mathcal{C}^2 \quad \text{since } \mathcal{C} > 2n\end{aligned}$$

where the equation from Lemma 1 is applicable, since $p = 1/\mathcal{C} < 1/(2n)$. Thus, expected cost is at most:

$$\begin{aligned}\left(d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})\right)\left(\frac{4n^2}{\mathcal{C}^2}\right)\mathcal{C} &= \frac{4dn^2\ln(w_{\text{cur}})}{\sqrt{\mathcal{C}}} \\ &< \frac{4dn\mathcal{C}\ln(w_{\text{cur}})}{\sqrt{\mathcal{C}}} \quad \text{since } \mathcal{C} > 2n \\ &= 4dn\sqrt{\mathcal{C}}\ln(w_{\text{cur}}) \\ &= 4dn\sqrt{\mathcal{C}}\ln(10n\sqrt{\mathcal{C}}) \quad \text{since } w_{\text{cur}} \in \text{LOW} \\ &= O(n\sqrt{\mathcal{C}}\ln(n)) \quad \text{since } \mathcal{C} = O(\texttt{poly}(n))\end{aligned}$$

as claimed. □

Lemma 17. *If $w_{\text{cur}} \in$ UNCERTAIN-LOW $\cup$ GOOD $\cup$ UNCERTAIN-HIGH, then the execution of* COLLECT-SAMPLE *has $O(\sqrt{\mathcal{C}}\ln(n))$ expected collision cost.*

Proof. By Lemma 1 with $m = n$ and $p \leq \frac{1}{n\sqrt{\mathcal{C}}}$, the probability of a collision is at most:

$$\begin{aligned}\frac{2m^2p^2}{1-mp} &\leq \frac{2n^2\frac{1}{n^2\mathcal{C}}}{1-\frac{n}{n\sqrt{\mathcal{C}}}} \quad \text{since this value is largest for } p = \tfrac{1}{n\sqrt{\mathcal{C}}} \\ &= \frac{2}{\mathcal{C}\left(1-\frac{1}{\sqrt{\mathcal{C}}}\right)} \\ &= O(1/\mathcal{C})\end{aligned}$$

Let $X = \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i$. The expected number of collisions in the sample of $d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})$ slots is:

$$\begin{aligned} E[X] &\geq E\left[\sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i\right] \\ &= \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} E[X_i] \quad \text{by linearity of expectation} \\ &= O\left(\frac{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})}{\mathcal{C}}\right) \\ &= O\left(\frac{d\ln(n)}{\sqrt{\mathcal{C}}}\right) \quad \text{since } \mathcal{C} = \texttt{poly}(n) \end{aligned}$$

Thus, the expected collision cost is at most:

$$O\left(\frac{d\ln(n)}{\sqrt{\mathcal{C}}}\right)\mathcal{C} = O(\sqrt{\mathcal{C}}\ln(n))$$

as claimed. □

Lemma 18. *If $w_{cur} \in$ HIGH, the execution of* COLLECT-SAMPLE *results in an $O(\sqrt{\mathcal{C}}\ln(n))$ expected collision cost.*

Proof. For HIGH, by using Lemma 1 we plug in the value $m = n$ (number of packets) and $p \leq \frac{1}{10^5 n\sqrt{\mathcal{C}}}$ (sending probability). The probability of a collision is at most:

$$\begin{aligned} \frac{2m^2p^2}{1-mp} &\leq \frac{2n^2\frac{1}{n^2 10^{10}\mathcal{C}}}{1-\frac{n}{10^5 n\sqrt{\mathcal{C}}}} \quad \text{since this value is largest for } p = \tfrac{1}{10^5 n\sqrt{\mathcal{C}}} \\ &= \frac{2}{10^{10}\mathcal{C}\left(1-\frac{1}{10^5\sqrt{\mathcal{C}}}\right)} \\ &= \Theta\left(\frac{1}{\mathcal{C}}\right) \end{aligned}$$

We have a sample of size $d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})$. Let $X = \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i$. The expected number of collisions in the sample of $d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})$ slots is:

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} X_i\right] \\ &= \sum_{i=1}^{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})} E[X_i] \quad \text{by linearity of expectation} \\ &= O\left(\frac{d\sqrt{\mathcal{C}}\ln(w_{\text{cur}})}{C}\right) \\ &= O\left(\frac{d\ln(n)}{\sqrt{\mathcal{C}}}\right) \quad \text{since } \mathcal{C} = \texttt{poly}(n). \end{aligned}$$

Thus, multiplying by $\mathcal{C}$, we obtain that the expected collision cost is at most $O(\sqrt{\mathcal{C}}\ln(n))$. □

Lemma 19. *The executions of* COLLECT-SAMPLE *and* DIAGNOSIS *guarantee that: (i)* RUNDOWN *is executed within $O(\sqrt{\mathcal{C}}\log^2(n))$ slots, and (ii) the total expected collision cost until that point is $O\left(n\sqrt{\mathcal{C}}\log^2(n)\right)$.*

Proof. We first bound the number of slots, and then we bound the expected collision cost.

(i) Bound on Number of Slots. By Lemmas 9, 10, and 14, DIAGNOSIS correctly doubles the window or executes RUNDOWN. Similarly, by Lemmas 13 and 15, DIAGNOSIS correctly halves the window or executes RUNDOWN. Therefore, DIAGNOSIS is always making progress towards reaching GOOD.

How many window halvings or doublings are required to reach GOOD? We start at an initial window of size $\mathcal{C}$, and either GOOD is smaller—in which case, we must perform $O(\log(\mathcal{C})) = O(\log(n))$ samples—or larger—in which case, we must again perform $O(\log(n\sqrt{\mathcal{C}})) = O(\log(n))$ samples. Thus, the number of slots until GOOD is reached is at most the sample size multiplied by this number of samples, which is $O(\sqrt{\mathcal{C}}\log^2(n))$. Then, by Lemma 12, RUNDOWN is executed.

(ii) Bound on Expected Collision Cost. Let S_i denote the collision cost for sample i when executing DIAGNOSIS, where $i = 1, ..., h$, where $h = O(\log(n))$ is the number of samples. Denote the total cost from collisions by $S = \sum_{i=1}^{h} S_i$. By Lemmas 16, 17, and 18, we have:

$$\begin{aligned} E[S] &= O\left(h\, n\sqrt{\mathcal{C}}\log(n)\right) \\ &= O\left(n\sqrt{\mathcal{C}}\log^2(n)\right) \end{aligned}$$

as claimed. □

A.4 Correctness and Cost Analysis for RUNDOWN

A single execution of lines 33 to 35 in RUNDOWN of COLLISION-AVERSION BACKOFF is called a ***run***. In any run, all packets operate over a window of size w_{cur} and send themselves in each slot with probability $2/w_{\text{cur}}$. At the end of the window, those packets that succeeded will terminate, while the remaining packets will carry on into the next window, which has size $w_{\text{cur}}/2$.

Our overall goal is to show that (roughly) a logarithmic in n runs are sufficient for all packets to succeed. To this end, our next lemma (below) argues that in each run at least half of the remaining packets succeed. Again, throughout this section, our results hold with a tunable probability of error that is polynomially small in n, and we omit this in our lemma statements.

Lemma 20. *Suppose m packets execute a window of* RUNDOWN*(w_{cur}), where $\gamma \ln n \le m \le n$, $\gamma \ge 1$ is a positive constant, and $w_{cur} \ge 8m\sqrt{\mathcal{C}}$. Then, at least $m/2$ packets succeed.*

Proof. Fix a packet and calculate the probability of failing over the entire window of size $8m\sqrt{\mathcal{C}}$. To do so, note that the probability that this packet succeeds in a fixed slot is: (i) the probability that the packet sends in this slot, multiplied by (ii) the probability that all other $m-1$ packets remain silent. That is:

$$\left(\frac{2}{w_{\text{cur}}}\right)\left(1-\frac{2}{w_{\text{cur}}}\right)^{m-1}.$$

Thus, the probability of failure in the slot is:

$$1-\left(\frac{2}{w_{\text{cur}}}\right)\left(1-\frac{2}{w_{\text{cur}}}\right)^{m-1}.$$

It follows that the probability of failing over all w_{cur} slots is:

$$\begin{aligned}
\left(1-\frac{2}{w_{\text{cur}}}\left(1-\frac{2}{w_{\text{cur}}}\right)^{m-1}\right)^{w_{\text{cur}}} &\leq \left(1-\frac{2}{w_{\text{cur}}}\frac{1}{e^{\frac{4(m-1)}{w_{\text{cur}}}}}\right)^{w_{\text{cur}}} \quad \text{by Fact 1(c)} \\
&\leq \left(1-\frac{2}{w_{\text{cur}}}\frac{1}{e^{\frac{4m}{w_{\text{cur}}}}}\right)^{w_{\text{cur}}} \\
&\leq e^{-\frac{2w_{\text{cur}}}{(w_{\text{cur}})(e^{4m/w_{\text{cur}}})}} \quad \text{by Fact 1(a)} \\
&\leq e^{-\frac{2}{e^{4m/w_{\text{cur}}}}} \\
&\leq \frac{1}{e^{\frac{2}{e^{4m/w_{\text{cur}}}}}} \\
&\leq \frac{1}{e^{\frac{2}{e^{4m/(8m\sqrt{\mathcal{C}})}}}} \quad \text{since this is maximized when } w_{\text{cur}} = 8m\sqrt{\mathcal{C}} \\
&\leq \frac{1}{e^{\left(\frac{2}{e^{1/(2\sqrt{\mathcal{C}})}}\right)}} \\
&\leq 0.3 \text{ for all } \mathcal{C} \geq 1. \qquad (4)
\end{aligned}$$

For $i = 1, ..., m$, let the indicator random variable $X_i = 1$ if packet i fails over the entire window; otherwise, let $X_i = 0$. We let $X = \sum_{i=1}^{m} X_i$, and note that:

$$\begin{aligned}
E[X] &\leq E\left[\sum_{i=1}^{m} X_i\right] \\
&\sum_{i=1}^{m} E[X_i] \quad \text{by linearity of expectation} \\
&= \sum_{i=1}^{m} Pr(X_i = 1) \\
&\leq 0.3m
\end{aligned}$$

By Theorem 3, letting $\delta = 2/3$, we have:

$$\begin{aligned}
Pr\left(X \geq (1+\delta)0.3m\right) &= Pr\left(X \geq m/2\right) \\
&\leq e^{-(2/3)^2 0.3m/(8/3)} \\
&= e^{-m/20} \\
&\leq e^{-(\gamma/20)\ln n} \quad \text{since } m \geq \gamma \ln n \\
&= 1/n^{d'}.
\end{aligned}$$

Therefore, with probability at least $1 - 1/n^{d'}$, at least half of the packets succeed. □

Discussion of Lemma 20. We highlight that Lemma 20 applies to all windows in a fixed run. That is, in the first window of the run we have $m = n$ and $w_{\text{cur}} = kn\sqrt{\mathcal{C}}$ for some constant $k \geq 8$. In the second window of the run, we have $m \leq n/2$ and $w_{\text{cur}} = k(n/2)\sqrt{\mathcal{C}}$. Referring to Line 32 of COLLISION-AVERSION BACKOFF, we see that this process—whereby the number of packets is *at least* halved, and the window size is halved—stops once we reach a window in the run with size less than $\Theta(\lg(w_0)\sqrt{\mathcal{C}})$, where $w_0 = \Theta(n\sqrt{\mathcal{C}})$.

How many windows are in the run? That is, how many times can we halve our initial window of size $w_0 = kn\sqrt{\mathcal{C}}$ before it drops below size $\lg(w_0)\sqrt{\mathcal{C}} = \lg(kn\sqrt{\mathcal{C}})\sqrt{\mathcal{C}}$? We solve for the number of runs j in:

$$\frac{kn\sqrt{\mathcal{C}}}{2^j} = \lg\left(kn\sqrt{\mathcal{C}}\right)\sqrt{\mathcal{C}}$$

which yields $j = \lg(kn) - \lg\lg(kn\sqrt{\mathcal{C}})$. Since $\mathcal{C} = \texttt{poly}(n)$, this means $j = \lg(n) - \lg\lg(n) + O(1)$. This implies that, over these j windows in the run, n packets will be reduced to no more than $n/2^j = O(\log n)$ packets by the final window. Therefore, we have the following corollary.

Corollary 1. *The number of windows in the run in* RUNDOWN *is* $\lg(n) - \lg\lg(n) + O(1)$*, after which only* $O(\log(n))$ *packets remain.*

For the remainder of this section, all of the lemmas that we invoke hold with high probability in n, and thus we omit this from most of our lemmas statements and arguments. We are now able to prove correctness and an upper bound on the number of slots incurred by RUNDOWN.

Lemma 21. *When* RUNDOWN *is executed, all packets succeed within* $O(n\sqrt{\mathcal{C}}\ln(n))$ *slots.*

Proof. By Lemmas 12, 14, and 15, RUNDOWN will execute with a window size $w \in [10n\sqrt{\mathcal{C}}, 10^5 n\sqrt{\mathcal{C}}]$. By Corollary 1, RUNDOWN reduces the number of remaining packets to $O(\log n)$ and uses $r = \lg(n) - \lg\lg(n) + O(1)$ windows in the run to do so. Thus, the number of slots RUNDOWN executes over is at most:

$$\sum_{j=0}^{r} \frac{10^5 n\sqrt{\mathcal{C}}}{2^j} = O\left(n\sqrt{\mathcal{C}}\right)$$

by the sum of a geometric series.

Any remaining packets, of which there are $O(\log n)$, execute over a window of size $w_0 \in [10n\sqrt{\mathcal{C}}, 10^5 n\sqrt{\mathcal{C}}]$. By the same analysis used to reach Equation 4 in the proof of Lemma 20, each packet succeeds with probability at least 0.7.

Repeating this $c\ln(w_0) \leq c\ln(n)$ times guarantees that all remaining packets succeed with an error bound of at most:

$$\begin{aligned}\left(\frac{7}{10}\right)^{c\ln(n)} &\leq n^{c\ln(7/10)} \\ &\leq \frac{1}{n^{d'}}\end{aligned}$$

so long as c is sufficiently large. Finally, the number of slots incurred by this component of RUNDOWN is $O(\log^2(n))$. □

The next lemma establishes an upper bound on the expected collision cost for RUNDOWN.

Lemma 22. *The expected collision cost for executing* RUNDOWN *is* $O(n\sqrt{\mathcal{C}}\ln(n))$.

Proof. By Lemmas 12, 14, and 15, RUNDOWN first executes with $m = n$ and $w_0 \in [10n\sqrt{\mathcal{C}}, 10^5 n\sqrt{\mathcal{C}}]$. By Lemma 20, in each window of a run, the number of packets is reduced by at least a factor of 2, and then the window size halves. That is, at the start of the i-th run, the number of packets is $m_i \leq n/2^i$ and the window size is $w_i = w_0/2^i$, for $i \geq 0$.

We employ Lemma 1, where $p_i = 2/w_i$. We have that the expected number of collisions over a window of size w_i in the run is at most:

$$\begin{aligned}\left(\frac{2m_i^2 p_i^2}{1 - m_i p_i}\right) w_i &= \left(\frac{2m_i^2 p_i^2}{1 - m_i p_i}\right)\left(\frac{2}{p_i}\right) \\ &= \frac{4m_i^2 p_i}{1 - m_i p_i} \\ &= O(m_i^2 p_i)\end{aligned}$$

By Corollary 1, there are $r = \lg(n) - \lg\lg(n) + O(1)$ windows in the run. Summing up over these windows, by linearity of expectation, the total expected number of collisions is:

$$\begin{aligned} O\left(\sum_{i=0}^{r} m_i^2 p_i\right) &= O\left(\sum_{i=0}^{r} \left(\frac{n}{2^i}\right)^2 \left(\frac{2}{w_i}\right)\right) \\ &= O\left(\sum_{i=0}^{r} \left(\frac{n}{2^i}\right)^2 \left(\frac{2^{i+1}}{n\sqrt{\mathcal{C}}}\right)\right) \\ &= O\left(\sum_{i=0}^{r} \left(\frac{n}{2^i\sqrt{\mathcal{C}}}\right)\right) \\ &= O\left(\frac{n}{\sqrt{\mathcal{C}}} \sum_{i=0}^{r} \frac{1}{2^i}\right) \\ &= O\left(\frac{n}{\sqrt{\mathcal{C}}}\right) \end{aligned}$$

where the last line follows from the sum of a geometric series. Thus, multiplying this by $\mathcal{C}$, we obtain that the expected collision cost is $O(n\sqrt{\mathcal{C}})$.

Finally, in lines 36 to 38, the remaining $O(\log n)$ packets execute over a window of size w_0, doing so $O(\ln(w_0) = O(\ln(n))$ times before succeeding, as established by Lemma 21. Certainly, each of the $O(\ln(w_0))$ executions has expected collision cost no more than that incurred by lines 32 to 35 (where we have n packets executing), and thus the expected collision cost is $O(n\sqrt{\mathcal{C}}\ln(w_0)) = O(n\sqrt{\mathcal{C}}\ln(n))$. □

A.5 Analysis of COLLISION-AVERSION BACKOFF

We wrap up our analysis of CAB by calling on our main lemmas from the previous sections.

Theorem 1 *W.h.p.* COLLISION-AVERSION BACKOFF *guarantees that all packets succeed, the makespan is* $O(n\sqrt{\mathcal{C}}\log(n))$*, and the expected collision cost is* $O(n\sqrt{\mathcal{C}}\log^2(n))$.

Proof. By Lemma 19, over the course of executing COLLECT-SAMPLE and DIAGNOSIS, w.h.p. CAB executes over $O(\sqrt{\mathcal{C}}\log^2(n))$ slots and the expected collision cost is $O(n\sqrt{\mathcal{C}}\log^2(n))$. By Lemmas 21 and 22, w.h.p. when RUNDOWN is executed in CAB, all packets succeed within $O(n\sqrt{\mathcal{C}}\ln(n))$ slots and the expected collision cost is $O(n\sqrt{\mathcal{C}}\ln(n))$. Therefore, CAB guarantees that all packets succeed, the makespan is $O(n\sqrt{\mathcal{C}}\log(n))$, and the expected collision cost is $O(n\sqrt{\mathcal{C}}\log^2(n))$. □

B Lower Bound

We consider only the set of slots, $\boldsymbol{\mathcal{S}}$, in the execution of any algorithm where at least two active packets remain, since we cannot have a collision with a single active packet. While we do not always make this explicit, but going forward, any slot t is assumed to implicitly belong to $\mathcal{S}$.

Let $\boldsymbol{p_i(t)}$ denote the probability that packet i sends in slot t. Note that, if a packet has terminated, its sending probability can be viewed as 0. For any fixed slot t, the ***contention*** in slot t is $\mathtt{Con}(t) = \sum_{i=1}^{n} p_i(t)$; that is, the sum of the sending probabilities in that slot.

We begin by arguing that any algorithm with contention exceeding 2 in any slot must incur an expected collision cost of $\Omega(\mathcal{C})$. We do this by deriving an upper bound on the probability of (i) a success and (ii) an empty slot, as a function of contention. In turn, this provides a lower bound on the probability of a collision that is useful when contention exceeds 2, allowing us to show $\Omega(\mathcal{C})$ expected cost in Lemma 25.

Lemma 23. *Fix any slot* t*. The probability of a success is at most* $\frac{e \cdot Con(t)}{e^{Con(t)}}$.

Proof. Fix a slot t. For any sending probabilities $p_i(t)$, the probability of a success in slot t is:

$$\begin{aligned}
&\sum_{\text{all } j}\left(p_j(t)\prod_{\text{all } i\neq j}(1-p_i(t))\right)\\
&\leq \sum_{\text{all } j}\left(p_j(t)e^{-\sum_{\text{all } i\neq j}p_i(t)}\right) \quad \text{by Fact 1(a)}\\
&\leq \sum_{\text{all } j}\left(p_j(t)e^{-(\texttt{Con}(t)-1)}\right) \quad \text{since } p_j(t)\leq 1\\
&\leq \frac{\texttt{Con}(t)}{e^{\texttt{Con}(t)-1}}\\
&\leq \frac{e\texttt{Con}(t)}{e^{\texttt{Con}(t)}}
\end{aligned}$$

which completes the proof. □

Lemma 24. *Fix any slot t. If $\texttt{Con}(t) > 2$, then the probability of a collision is at least $1/10$.*

Proof. By Lemma 23, the probability of a success in slot t is at most $\frac{e\texttt{Con}(t)}{e^{\texttt{Con}(t)}}$. The probability that slot t is empty is at most $\prod_{\text{all j}}(1-p_j(t)) \leq e^{-\texttt{Con}(t)}$ by Fact 1(a). Therefore, the probability of a collision is at least:

$$\begin{aligned}
1-\frac{e\,\texttt{Con}(t)}{e^{\texttt{Con}(t)}}-e^{-\texttt{Con}(t)} &\geq 1-\frac{2e}{e^2}-e^{-2} \quad \text{since } \texttt{Con}(t)>2\\
&\geq 1/10.
\end{aligned}$$

as claimed. □

Lemma 25. *Any algorithm that has $\texttt{Con}(t) > 2$ has an expected collision cost that is $\Omega(\mathcal{C})$.*

Proof. Let t be any slot where $\texttt{Con}(t) > 2$ in the execution of any algorithm. By Lemma 24, the probability of a collision is at least $1/10$. Therefore, the expected cost from collisions is at least $\mathcal{C}/10$. □

Next, we need to consider algorithms with "low" contention in every slot; that is, $\texttt{Con}(t) \leq 2$ for all t. As a stepping stone, in Lemmas 26 we derive a lower bound on the probability of a collision that is a function of contention, and which is easy to work with when contention is between 0 and 2.

Lemma 26. *Fix any slot t and let $\texttt{Con}(t) \leq 2$. The probability of a collision in t is at least $(\frac{1}{110})\left(\texttt{Con}(t)^2-\sum_i p_i(t)^2\right)$.*

Proof. For ease of presentation, in this proof we write p_i instead of $p_i(t)$. We restrict our analysis to collisions involving only two packets; note that considering collisions involving three or more packets can only increase the probability of a collision.

For the following analysis, we restrict all sending probabilities p_i are at most $1/2$; however, we will remove this restriction momentarily. The probability of collision in slot t is at least:

$$\begin{aligned}
\frac{1}{2}\Bigg(&p_1p_2\prod_{i=1}^{n}(1-p_i)+p_1p_3\prod_{i=1}^{n}(1-p_i)+\ldots+p_1p_n\prod_{i=1}^{n}(1-p_i)+\\
&p_2p_1\prod_{i=1}^{n}(1-p_i)+p_2p_3\prod_{i=1}^{n}(1-p_i)+\ldots+p_2p_n\prod_{i=1}^{n}(1-p_i)+\\
&\vdots\\
&p_np_1\prod_{i=1}^{n}(1-p_i)+p_np_2\prod_{i=1}^{n}(1-p_i)+\ldots+p_np_{n-1}\prod_{i=1}^{n}(1-p_i)\Bigg)
\end{aligned}$$

where the leading factor of $1/2$ comes from double counting terms (e.g., p_1p_2 and p_2p_1), which allows us to simply below. Since each sending probability is at most $1/2$, we can use Fact 1(c) to simplify in the following way.

$$\begin{aligned}
&\geq \left(\frac{1}{2}\right) e^{-2\texttt{Con}(t)} \left(p_1 \sum_{j\neq 1} p_j + p_2 \sum_{j\neq 2} p_j + \cdots + p_n \sum_{j\neq n} p_j\right) \\
&= \left(\frac{1}{2}\right) e^{-2\texttt{Con}(t)} \left(p_1\left(\left(\sum_{\text{all } j} p_j\right) - p_1\right) + p_2\left(\left(\sum_{\text{all } j} p_j\right) - p_2\right) + \cdots + p_n\left(\left(\sum_{\text{all } j} p_j\right) - p_n\right)\right) \\
&= \left(\frac{1}{2}\right) e^{-2\texttt{Con}(t)} \left(\left(p_1\texttt{Con}(t) - p_1^2\right) + \left(p_2\texttt{Con}(t) - p_2^2\right) + \cdots + \left(p_n\texttt{Con}(t) - p_n^2\right)\right) \\
&= \left(\frac{1}{2}\right) e^{-2\texttt{Con}(t)} \left(\texttt{Con}(t)(p_1 + p_2 + \cdots + p_n) - \sum p_j^2\right) \\
&\geq \frac{\texttt{Con}(t)^2 - \sum_i p_i^2}{2e^{2\texttt{Con}(t)}} \\
&\geq \frac{\texttt{Con}(t)^2 - \sum_i p_i^2}{110} \quad \text{since } \texttt{Con}(t) \leq 2.
\end{aligned}$$

Finally, note that if we allow the sending probabilities to be larger than $1/2$, this can only increase the probability of a collision; therefore, our lower bound holds even when the sending probabilities are not restricted. □

Before making our next argument, consider a simple example with two packets, whose sending probabilities are $p_1(t) = 1$ and $p_2(t) = 1/100$ in slot t. Observe that $\texttt{Con}(t)^2 - (p_1^2 + p_2^2) = 1/50 \ll \texttt{Con}(t)^2$. However, here $\Delta(t) = 1/100$, and it is true that $\texttt{Con}(t)^2 - (p_1^2 + p_2^2) \geq \Delta(t)\texttt{Con}(t)^2$. Lemma 27 below generalizes this inequality.

Lemma 27. *For any fixed slot t, $\texttt{Con}(t)^2 - \sum_i p_i(t)^2 \geq \Delta(t)\ \texttt{Con}(t)^2/2$.*

Proof. For ease of presentation, in this proof we write p_i instead of $p_i(t)$. Fix a slot t. Without loss of generality let the summands of $\texttt{Con}(t)$ be labeled in non-monotonic decreasing order: $p_1 \geq p_2 \geq ... \geq p_n$. For part one of our argument, we note that:

$$p_1p_2 \geq p_1^2\Delta(t) \tag{5}$$

since $\Delta(t) = p_2/p_1$.

Part two of our argument proceeds as follows. For $i \geq 2$, for any p_i^2 term removed from $\texttt{Con}(t)^2$, we focus on a 'leftover" term in $\texttt{Con}(t)^2 - \sum_i p_i^2$ of the form $p_{i-1}p_i$. For example, for p_2^2 we have a leftover term p_1p_2, and we note that $p_1p_2 \geq p_2^2$, since $p_1 \geq p_2$. Generalizing, the leftover term $p_{i-1}p_i$ satisfies $p_{i-1}p_i \geq p_i^2$, since $p_{i-1} \geq p_i$.

We tie things together by first noting:

$$\texttt{Con}(t)^2 - \sum_i p_i^2 \geq p_1p_2 + p_1p_2 + p_2p_3 + p_3p_4 + ... + p_{n-1}p_n$$

where we highlight that we need both p_1p_2 terms; one to offset the subtraction of p_1^2 from $\texttt{Con}(t)$, as discussed in part one of our argument, and the other to offset the subtraction of p_2^2 from $\texttt{Con}(t)$, as discussed in part two of our argument. For the subtraction of p_i^2 for $i \geq 3$, we need only a single corresponding term $p_{i-1}p_i$ to offset this. Thus, continuing from above, we have:

$$\begin{aligned}
&p_1p_2 + p_1p_2 + p_2p_3 + p_3p_4 + ... + p_{n-1}p_n \\
&\geq p_1^2\Delta(t) + p_1p_2 + p_2p_3 + p_3p_4 + ... + p_{n-1}p_n \quad \text{by Equation 5} \\
&\geq p_1^2\Delta(t) + p_2^2 + p_3^2 + ... + p_n^2 \quad \text{since } p_{i-1}p_i \geq p_i^2 \text{ for } i \geq 2 \\
&\geq \Delta(t)\sum_i p_i^2 \quad \text{since } \Delta(t) \leq 1
\end{aligned} \tag{6}$$

If $\sum_i p_i^2 \geq \texttt{Con}(t)^2/2$, then by the above:

$$\begin{aligned}
\texttt{Con}(t)^2 - \sum_i p_i^2 &\geq \Delta(t)\sum_i p_i^2 \quad \text{by Equation 6} \\
&\geq \Delta(t)\,\texttt{Con}(t)^2/2.
\end{aligned}$$

Else, $\sum_i p_i^2 < \texttt{Con}(t)^2/2$ and $\texttt{Con}(t)^2 - \sum_i p_i^2 \geq \texttt{Con}(t)^2/2 \geq \Delta(t)\texttt{Con}(t)^2/2$, given $\Delta(t) \leq 1$. Either way, the claim follows. □

For any algorithm $\mathcal{A}$ that executes over a set of slots $\mathcal{I}$, we define ***total contention*** of $\mathcal{A}$ to be $\sum_{s\in\mathcal{I}} \texttt{Con}(t)$; that is, the sum of contention over all slots of $\mathcal{A}$'s execution. Note, for this definition, we are counting those slots during which a single packet is active (for simplicity, we do not discuss this intermediate result in the overview of Section 4, so we were able to restrict our attention to only the set $\mathcal{S}$).

Lemma 28. *Any algorithm that w.h.p. has n packets succeed has the property that the total contention over all slots in the execution is at least $n/16$.*

Proof. Assume the existence of an algorithm $\mathcal{A}$ that guarantees with probability at least $1 - 1/n$ that all n packets succeed, but has the property that the sum of contention over all slots of the execution is less than $n/16$.

For any packet u, define the *lifetime contribution* of u to be the sum of u's sending probabilities over all slots of the execution of $\mathcal{A}$; denote this by $\boldsymbol{LC(u)}$. Call packet u "heavy" if $LC(u) \geq 1/4$; otherwise, u is "light".

There must be less than $n/4$ heavy packets. Otherwise, the sum of the contention for all heavy packets is at least $(n/4)(1/4) \geq n/16$, which contradicts the existence of $\mathcal{A}$. Therefore, at least $(3/4)n$ packets are light.

Consider any light packet u. Let $p(t)$ denote the probability of that u sends in slot t specified arbitrarily by $\mathcal{A}$. The probability that u succeeds in slot t is at most $p(t)$; therefore, the probability that u fails in slot t is at least $1 - p(t)$. Over the execution of $\mathcal{A}$, the probability that u does not succeed is at least:

$$\begin{aligned}\prod_t (1 - p(t)) &\geq e^{-2\sum_t p(t)} \quad \text{by Fact 1(c)} \\ &\geq e^{-2\cdot LC(u)}.\end{aligned}$$

Since $LC(u) < 1/4$, then the probability of failure is at least:

$$\begin{aligned}e^{-2LC(u)} &\geq 1 - 2\,LC(u) \quad \text{by Fact 1(a)} \\ &\geq 1 - 2(1/4) \\ &\geq 1/2.\end{aligned}$$

Thus, there is a packet that with probability at least $1/2$ does not succeed over the execution of $\mathcal{A}$, which is a contradiction. □

Lemma 29. *Consider any algorithm that guarantees w.h.p. that all packets succeed. Consider the portion of the execution during which only one active packet remains. W.h.p., the sum of the contention over this portion of the execution is $O(\log n)$.*

Proof. Let u be the last packet to succeed. Let $\mathcal{I}$ be the set of slots during which u is the only remaining active packet. Let $p(t)$ denote the probability of that u sends in slot $t \in \mathcal{I}$ specified arbitrarily by the algorithm. Since it is the only active packet, the probability that u succeeds in slot t is $p(t)$; therefore, the probability that u fails in slot t is $1 - p(t)$. Over the execution of $\mathcal{A}$, the probability that u does not succeed is at most:

$$\prod_{t\in\mathcal{I}} (1 - p(t)) \leq e^{-\sum_{t\in\mathcal{I}} p(t)} \quad \text{by Fact 1(a)}$$

Note that for $\sum_{t\in\mathcal{I}} p(t) = O(\log n)$, this probability of failure is polynomially small in n. □

Lemma 30. *Any algorithm that guarantees w.h.p. that n packets succeed must w.h.p. have $\sum_{t\in\mathcal{S}} \texttt{Con}(t) = \Omega(n)$.*

Proof. By definition, $\sum_{t\in\mathcal{S}} \texttt{Con}(t)$ is at least the total contention in Lemma 28 minus the sum of the contention in Lemma 29, which is at least: $(n/16) - O(\log n) = \Omega(n)$. □

Lemma 31. *Consider any algorithm $\mathcal{A}$ whose contention in any slot is at most 2 and guarantees w.h.p. a makespan of $\tilde{O}(n\sqrt{C})$. W.h.p. the expected collision cost for $\mathcal{A}$ is $\tilde{\Omega}(\Delta_{\min} n\sqrt{C})$.*

Proof. Let X be a random variable that is $|\mathcal{S}|$ under the execution of $\mathcal{A}$. Note that, since w.h.p. the makespan is $O(n\sqrt{\mathcal{C}})$, it must be the case that w.h.p. $X = O(n\sqrt{\mathcal{C}})$.

By Lemma 30, we have:

$$\sum_{t=1}^{X} \texttt{Con}(t) \geq cn. \tag{7}$$

for some constant $c > 0$. Let $Y_t = 1$ if slot $t \in \mathcal{S}$ has a collision; otherwise, $Y_t = 0$. By Lemmas 26 and 27:

$$\begin{aligned} Pr(Y_t = 1) &\geq \Delta(t) \cdot \texttt{Con}(t)^2/220 \\ &\geq \Delta_{\min} \cdot \texttt{Con}(t)^2/220 \end{aligned}$$

where the second line follows from the definition of $\Delta_{\min}$. The expected collision cost is:

$$\begin{aligned} \sum_{t=1}^{X} P(Y_t = 1) \cdot \mathcal{C} &\geq \sum_{t=1}^{X} \frac{\Delta_{\min}\texttt{Con}(t)^2 \cdot \mathcal{C}}{220} \\ &= \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \sum_{t=1}^{X} \texttt{Con}(t)^2. \end{aligned} \tag{8}$$

By Jensen's inequality for convex functions, we have:

$$\frac{\sum_{t=1}^{X} \texttt{Con}(t)^2}{X} \geq \left(\frac{\sum_{t=1}^{X} \texttt{Con}(t)}{X} \right)^2 \tag{9}$$

Finally, the expected cost is at least:

$$\begin{aligned} \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \sum_{t=1}^{X} \texttt{Con}(t)^2 &\geq \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \frac{\left(\sum_{t=1}^{X} \texttt{Con}(t)\right)^2}{X} \quad \text{by Equations 8 and 9} \\ &\geq \frac{\Delta_{\min} \cdot \mathcal{C}}{220} \left(\frac{c^2 n^2}{X} \right) \quad \text{by Equation 7} \\ &= \tilde{\Omega}\left(\Delta_{\min}\, n\sqrt{\mathcal{C}} \right) \end{aligned}$$

where the second line follows by Equation 7, which was defined with regard to $\mathcal{S}$, and so can be compared to Equation 8. The last line follows since w.h.p. $X = \tilde{O}(n\sqrt{\mathcal{C}})$. □

We now state our lower bound, which follows directly from Lemmas 25 and 31.

Theorem 4. *Consider any algorithm that w.h.p. guarantees $\tilde{O}(n\sqrt{\mathcal{C}})$ makespan. Then, w.h.p., the expected collision cost for $\mathcal{A}$ is $\tilde{\Omega}(\min\{\mathcal{C}, \Delta_{\min} n\sqrt{\mathcal{C}}\})$.*

Addressing Theorem 2, note that for fair algorithms, $\Delta_{\min} = 1$. Recall that our analysis ignores any slot t where all packets have sending probability 0 (i.e.,$\texttt{Con}(t) = 0$), giving such slots to the algorithm for free. Given that $\Delta_{\min} = 1$, $\tilde{\Omega}(\min\{\mathcal{C}, \Delta_{\min} n\sqrt{\mathcal{C}}\}) = \tilde{\Omega}(n\sqrt{\mathcal{C}})$ when $\mathcal{C} = c' n^2$ for a sufficiently large positive constant c'.