SIAM J. COMPUT.
Vol. 47, No. 5, pp. 1735–1754

CONTENTION RESOLUTION WITH CONSTANT THROUGHPUT AND LOG-LOGSTAR CHANNEL ACCESSES*

MICHAEL A. BENDER†, TSVI KOPELOWITZ‡, SETH PETTIE‡, AND MAXWELL YOUNG§

Abstract. For decades, randomized exponential backoff has provided a critical algorithmic building block in situations where multiple devices seek access to a shared resource. Despite this history, the performance of standard exponential backoff is poor under worst-case scheduling of demands on the resource: (i) subconstant throughput can occur under plausible scenarios, and (ii) each of N devices requires $\Omega(\log N)$ access attempts before obtaining the resource. In this paper, we address these shortcomings by offering a new backoff protocol for a shared communication channel that guarantees expected constant throughput with only $O(\log(\log^* N))$ channel accesses in expectation, even when packet arrivals are scheduled by an adversary. Central to this result are new algorithms for *approximate counting* and *leader election* with the same performance guarantees.

Key words. contention resolution, distributed computing, algorithms, wireless networks, throughput, adversarial scheduling

AMS subject classifications. 68W15, 68W20, 68W40, 68M12

DOI. 10.1137/17M1158604

1. Introduction.

Randomized exponential backoff [59] is a classic algorithm for resolving contention when there is a collection of devices that all need to broadcast on a channel, but only one node can broadcast at a time.

This channel may correspond to a wireless scenario for which there exist a number of communication standards that govern access to the shared communication medium; a ubiquitous example is the IEEE 802.11 family of standards [1]. More generally, this scenario may model any shared resource for which devices require temporary exclusive access. For instance, randomized backoff is implemented in a broad range of applications including transactional memory [52], lock acquisition [69], email retransmission [18, 35], congestion control (e.g., TCP) [61, 53], and a variety of cloud computing applications [46, 66, 76].

In this paper, we design a substitute for randomized exponential backoff that is globally efficient (in terms of throughput), locally efficient (in terms of per device costs), and impervious to adversarially scheduled inputs.

1.1. The multiple-access channel model.

In classic algorithmic analyses of randomized backoff, there are N indistinguishable *players* that arrive over time, each

*Received by the editors November 29, 2017; accepted for publication (in revised form) August 6, 2018; published electronically October 2, 2018. A preliminary version of our results appeared as an extended abstract at STOC'16, ACM, New York, pp. 499–508 [14].

http://www.siam.org/journals/sicomp/47-5/M115860.html

Funding: This research was partially supported by National Science Foundation grants CCF-1217708, IIS-1247726, IIS-1251137, CNS-1408695, CCF-1439084, CCF-1217338, CNS-1318294, CCF-1514383, CNS-1318294, CCF-1613772, CCF-1815316, CCF-1617618, CCF-1763680, CCF-1716252, CCF-1725543, and CNS-1755615. Support is also provided by Sandia National Laboratories, NetApp, and a research gift from C Spire.

†Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-2424 (bender@cs.stonybrook.edu).

‡Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109 (kopelot@gmail.com, pettie@umich.edu).

§Department of Computer Science and Engineering, Mississippi State University, Mississippi State, MS 39759 (myoung@cse.msstate.edu).

of which needs to transmit a *packet* on a so-called *multiple-access channel*.[1] Time is divided into discrete *slots*. We assume that in each time slot we can convey a single packet of at least $\log N$-bits.[2]

We assume an *adaptive* adversary, which controls precisely how many players are injected into the system in each slot. In making its decisions, the adaptive adversary has access to the entire state of the system so far, the internal state of each player, but not the outcomes of future coin tosses. There is no universal numbering scheme for the slots, that is, no global clock from which a newly injected player can infer the lifetime of the system, slot parity, etc.

In each time slot, each player in the system can do one of four *actions*: (i) sleep, (ii) send their packet, (iii) send a packet-sized message—a $\log N$-bit number, or (iv) listen to the channel. Players who take actions (ii), (iii), or (iv) are said to *access* the channel. If no players send, then the slot is *empty*; if two or more players send, the slot is *full* and *noisy*; if *exactly* one player sends, then the slot is *full* and the transmission is *successful*. All players who access the channel (that is, commit actions (ii), (iii), and (iv)) can distinguish between an empty slot, a full and noisy slot, and a full slot where the transmission is successful; moreover, if the transmission is a successful one of type (iii), all listeners learn the $\log N$-bit number. A player is allowed to exit the system only after its packet has been successfully transmitted.

There are several axes along which we can evaluate a protocol. A global measure of effectiveness is *utilization*, defined as the ratio of the number of successful slots and N. The reciprocal of utilization is called *throughput*.

From a single player's perspective, there is some nonzero cost to access the channel (rather than take action (i), sleeping) and depending on the technology, the cost of sending (actions (ii) and (iii)) may be different from the cost of listening (action (iv)). Our goal is to design a protocol that guarantees constant expected utilization and that minimizes the expected number of channel accesses per player.

Our second objective (access cost) is meaningful in the multitude of applications where the channel-access cost is high enough to be worth conserving. For example, in a wired network, an unsuccessful transmission wastes bandwidth. In a wireless network, in which devices are battery powered, sending and listening takes energy and reduces battery life. In transactional memory, a transaction rollback (i.e., an unsuccessful attempt) wastes CPU cycles.

1.2. Results. In this paper, we prove the following theorem.

THEOREM 1. *There exists a randomized contention-resolution protocol enabling N online players to transmit their packets on a multiple-access channel such that*

- *an expected constant fraction of the slots contain a successfully transmitted packet;*
- *each player sends on the channel $O(1)$ times in expectation;*
- *each player listens on the channel $O(\log(\log^* N))$ times in expectation.*

Moreover, these bounds hold even if the arrival times of the players are controlled by an adaptive adversary.

Theorem 1 contrasts with standard binary randomized exponential backoff, which, for unknown N, requires $\Omega(\log N)$ channel accesses; this lower bound is shown in [11].

[1] Although it is usually fine to conflate the players with their associated packets, we find it useful to distinguish them. For example, in our protocols, players are obliged to do more than *merely* guarantee that their packet is sent.

[2] The notation log indicates the logarithm base 2.

One of the new ideas in our algorithm is circuits for cheaply computing functions of binary data encoded as full/empty slots. Randomness is used inside our circuit evaluation protocol for load balancing, but the evaluation itself has zero probability of error. Theorem 2 summarizes this independent component. The following theorems hold for general $n \leq N$.

THEOREM 2. *Let C be a circuit with ℓ bits of input and c constant fan-in gates. There exists a protocol for n players on a multiple-access channel (n being unknown) for evaluating C, where the input is represented by ℓ specified slots (empty $= 0$, full $= 1$), whose sending and listening cost per player is $O(1 + \frac{c}{n})$ in expectation.*

We also give an extremely efficient protocol for estimating n and for leader election. Our results are consistent with the classic lower bound of Willard [81], which applies to time, not channel accesses.

THEOREM 3. *There exists a randomized protocol enabling n players with a synchronized start time to* (1) *estimate n to within a constant factor and* (2) *elect a leader, such that*

- *each player sends on the channel $O(1)$ times in expectation, and*
- *each player listens on the channel $O(\log(\log^* n))$ times in expectation;*
- *the protocol completes after $n^{o(1)}$ time.*

1.3. Related work. A preliminary version of this work appeared as an extended abstract in [14]. Here, we have provided additional proof details, an expanded description of prior relevant work, new subsections addressing applications of our results and practical considerations, and an extended discussion of open problems.

Subsequent to [14], Chang et al. [26] proved that Theorem 3 is optimal in the sense that $\Omega(\log(\log^* N))$ channel accesses are necessary for any protocol taking $\text{poly}(n)$ time. They provided a more general trade-off curve between time and channel accesses for leader-election algorithms, e.g., with $O(\log^{2+\epsilon} n)$ time it is possible to make just $O(\log(\epsilon^{-1} \log\log\log n))$ channel accesses.

One variant of exponential backoff can be described as selecting slots within *windows*. When a new player is injected into the system it partitions future time into consecutive windows of length $W_0, W_1, W_2, \ldots$. If, at the beginning of window W_i, the player has yet to transmit his packet successfully, it attempts to do so at a slot in W_i chosen uniformly at random.

Bender et al. [11] analyzed such backoff schemes more generally. The main takeaway messages from [11] are that no *monotone* backoff strategy (in which $W_{i+1} \geq W_i$) has constant throughput, but that when all the players start at once, a simple *nonmonotone* strategy called *sawtooth* does have constant throughput.[3] These backoff strategies only access the channel by sending, not listening, and their sending cost is $\Omega(\log N)$ in expectation. Anderton and Young [5] evaluate these backoff schemes using simulations of a wireless network.

Queuing theory arrivals. For many years, most of the analytic results on backoff assumed statistical queuing-theory models and focused on the question of what packet-arrival rates are stable (see [51, 45, 68, 43, 51, 44]). Interestingly, even with Poisson arrivals, there are better protocols than binary exponential backoff, such as polynomial backoff [51].

Another important notion is that of *saturated throughput*; this is roughly, the maximum throughput when each player always has a packet to be sent. This notion has been examined in [19, 77].

[3] The *backoff-backon* idea behind sawtooth has been discovered in other contexts [41, 49, 6].

The findings in this paper are pertinent to both results involving statistical queueing theory and saturated throughput because we guarantee constant utilization under adversarially scheduled packet arrivals.

Other arrival models. When all the packets begin at the same time (known as the *static case*), efficient protocols for sharing a multiple-access channel are known [6, 12, 48]. In contrast, there has been work on adversarial queueing theory, with an examination of the worst-case performance in the multiple-access model [11, 31, 3].

Algorithms that take feedback on the channel into account when deciding their next action are *adaptive*. In [56], the authors present a nonadaptive algorithm for sharing a channel in a setting where players have access to a global clock. The players are awakened at *adversarially scheduled times* (this is also referred to as the *dynamic* scenario), and players cannot detect when a collision occurs, though the channel does report whether a transmission was successful or not. In [37], the authors present both adaptive and nonadaptive algorithms for adversarially chosen arrival times, but without a global clock. Optimal algorithms are presented for the cases of a nonadaptive algorithm that knows "n" and an adaptive algorithm that has no knowledge of "n"; the case of a nonadaptive algorithm and unknown "n" is proved to be strictly harder.

First successful transmission/estimating N. Willard [81] addresses a contention-resolution problem in which the goal is to minimize the *first* moment that some player transmits successfully. Sharp time bounds of $\Theta(\log\log N)$ (in expectation) are proved when the N players begin at the same time. Nakano and Olariu [63] improved the failure probability of Willard's algorithm; theirs takes $O(\log\log N + \log f^{-1})$ time for failure probability f.

The closely related wake-up problem [29, 28, 32, 27, 57, 54] addresses how long it takes for a single transmission to succeed when packets arrive under the dynamic scenario. While the problem considered here also deals with dynamic arrivals, *all* packets must be transmitted successfully.

Leader election is also a related problem, and its complexity is much higher without collision detection. The *decay* algorithm [10] takes $O(\log N \log f^{-1})$ time, and this is known to be optimal [64]. Chang et al. [26] gave a variety of trade-offs between time and channel accesses, both with and without collision detection.

Finally, when the number of players is unknown, size-estimation techniques have been developed in the context of exponential backoff and variants [47, 22, 23, 20].

Adversarial fault tolerance. A number of elegant results exist on contention resolution when the channel is subject to (possibly malicious) noise. Awerbuch, Richa, and Scheideler [8] and Richa et al. [70, 71, 72] consider an adversary whose jamming is bounded within any sufficiently large window of time. Under the same adversarial model, Ogierman et al. [65] examine the problem under the well-known signal-to-interference-plus-noise ratio (SINR) communication model.

Anantharamu et al. [2] consider a multiple access channel where dynamic arrivals occur subject to an injection rate and a jamming rate. They define a (ρ, λ, b)-adversary who may inject $\rho|\tau| + b$ packets and jam at most $\lambda|\tau| + b$ slots within any contiguous segment τ consisting of $|\tau|$ slots, where $\lambda < 1$.

A recent result by Bender et al. [13] also addresses worst-case online packet arrivals and in the face of an unknown T noisy slots scheduled by an adaptive adversary. It achieves expected constant utilization with an expected polylog$(N + T)$ channel accesses.

Relationship to balanced allocations. Scalable backoff is closely related to balls-and-bins games [9, 33, 60, 73, 79], and the analysis of balls-and-bins often shows up in problems involving load balancing, resource allocation, and scheduling [15, 16, 17, 30]. Bins correspond to time slots and balls correspond to players. The objective is for each ball to land in its own bin; if several balls share the same bin, they are rethrown. The flow of time gets modeled by restrictions on when balls get thrown and where they may land. As mentioned earlier, listening during a slot corresponds to observing whether balls landed in a bin, and decoding an $O(\log N)$-bit number, if a single ball landed in the bin. Our results show that, remarkably, we can achieve an expected $O(\log(\log^* N))$ throws and bin observations, while still achieving a constant utilization.

Relationship to the beeping model. Portions of our algorithm rely on conveying information by having each player either emit noise or remain silent in a slot. This aligns with the *beeping model* introduced by Cornejo and Kuhn [34]. Under this model, both leader election [42] and counting [24, 21] have been investigated.

1.4. Algorithm overview. At any moment in a contention-resolution protocol we can measure the *contention* in a single slot by summing up the probabilities of each player sending in that slot. In order to have constant throughput, we need the contention on a constant fraction of the slots to be constant. If there are currently $n \leq N$ players in the system, a natural way to achieve constant contention is to have all players send with probability $\Theta(1/n)$, which requires that players estimate n, either implicitly (via some kind of backoff) or explicitly. The first difficulty is that n is unknown and unbounded. The second is that n is constantly changing: players successfully transmit their packets and leave the system, and the adversary injects new players into the system.

We give an efficient protocol for estimating n when the n players start at the same time, that is, they agree on a slot zero. Once all n players have a mutually agreed-upon estimate $\tilde{n}$ of n, they can skip *directly* to the correct iteration of the sawtooth algorithm and finish in $\Theta(\tilde{n})$ slots with probability $1 - 1/\operatorname{poly}(n)$, each player sending in $O(1)$ slots in expectation. The sawtooth is *truncated* on both sides: a prefix of the execution is skipped (because we know an estimate of n) and the algorithm ends after $\Theta(\tilde{n})$ slots regardless of whether a few packets still need to be transmitted.

The protocol outlined above demands perfect synchronization. Even if the players are out of step by just one time slot, the protocol could fail; therefore, in our algorithms, we develop a method to achieve synchronization. Although the players arrive at various times, they organize themselves into *batches*. In each batch, all players have an agreed-upon time zero, and they run a protocol for estimating n and then execute a truncated sawtooth.

The sawtooth contention resolution protocol of [11] is somewhat similar to the windowed version of binary exponential backoff, but is *non*monotonic. It consists of iterations indexed by $i \geq 0$, each of which consists of windows of length $2^i, 2^{i-1}, 2^{i-2}, \ldots$. It is fairly straightforward to show that at any iteration $i \geq \log n + O(1)$, all n packets are successfully transmitted, and are actually transmitted within the first $\log\log n + O(1)$ windows of iteration i with high probability. Thus, once we have an estimate $\tilde{n}$, we can jump to iteration $\log\tilde{n} + O(1)$ of the sawtooth protocol. By stopping after $\log\log\tilde{n} + O(1)$ windows, the worst-case sending cost per player is $O(\log\log\tilde{n})$, the expected sending cost is $O(1)$, and all n players finish with high probability in n. This is the *truncated sawtooth phase*.

Batches never run concurrently. Consider the point of view of a new player entering the system. Because of the nonconcurrency requirement, this player must quickly distinguish between two situations: (i) no batch is currently running, in which case one can be started, possibly with other players waiting in the system, or (ii) some batch is running, in which case the player must determine precisely when it will end in order to identify when a new batch will start. All the players in a batch will collectively spend a constant fraction of the slots *humming*, that is, making noise. Humming indicates that a batch is currently running and prevents new players from starting a concurrent batch.

The protocol components are given in more detail below.

The players are synchronized. Suppose that $n \leq N$ players enter the system at the same time and that they implicitly agree on time slot zero. The players begin with an *estimation phase*, where the goal is to compute a $\Theta(1)$-approximation $\tilde{n}$ of n collectively.

First we address the simpler problem of testing whether or not n is close to 2^i. In a series of $\Omega(i)$ slots, each player makes noise in each slot with probability 2^{-i}. If n is close to 2^i, Chernoff bounds imply that a large constant fraction of the slots are clear *and* a large constant fraction of the slots are noisy. In contrast, if n is far from 2^i, then the slots are either mostly noisy or mostly clear. One can construct a $\Theta(i)$-sized circuit C_i that counts the number of 1 inputs (noisy slots $= 1$, clear slots $= 0$) and compares this sum against a threshold, which implies (see Theorem 2) that the players can cheaply test whether or not n is close to 2^i.

Suppose that we have deduced that $n \geq X$. We build circuit testers that test n against powers of 2 from $2^{\log X}, 2^{1+\log X}, \ldots, 2^{X^{1/3}}$. The aggregate size of these circuits is $O(X^{2/3}) = O(n^{2/3})$, so they can all be simulated cheaply using Theorem 2. If no circuit outputs 1 then we have a better lower bound on n, namely, $n > 2^{X^{1/3}}$, and we can now afford to simulate a much larger collection of circuits in the next iteration.

On the other hand, if n is in the range $[X, 2^{X^{1/3}}]$, then some subset of the players will learn an estimate of $n \approx 2^i$, namely, the players that simulated the output gate of the circuit C_i. In order for this subset of players to notify everyone else that $n \approx 2^i$, they first elect a leader, using another $O(X^{1/3})$ slots. In a designated slot, the leader announces "2^i" and every other player listens. The players make $O(1)$ channel accesses in expectation to simulate a batch of circuits and elect a leader.

A sequential search for n would involve evaluating $\log^* n$ batches of circuits. To speed up this algorithm the n players initially perform a straightforward binary search for $\log^* n \pm 1$, which requires $O(\log(\log^* n))$ slots and channel accesses per player. This estimate of $\log^* n$ implies a decent enough lower bound on n that the players only need to simulate $O(1)$ batches of circuits. The estimation phase is described in detail in Section 4.

The process of estimating n may yield a significant underestimate or overestimate. Even if the estimate is accurate, a leader may fail to be elected. In either case, the probability of such a bad event is very small. These rare errors are eventually detected and remedied by the relevant players. For example, if a leader fails to be elected, they can restart the estimation phase. These restarts do not change the players' expected cost asymptotically.

Batches and humming. Players arrive online and must be organized into synchronized *batches*. Each batch is a set of players that execute the estimation and

truncated sawtooth phases, which together define one *epoch*. Players in the current batch are called *active*. Our algorithm guarantees (via a humming mechanism described shortly) that *inactive* players injected into the system in the middle of an epoch quickly learn that fact and eventually learn precisely when the epoch ends so that they may become active in the next batch. Thus, epochs are disjoint. If a player does not succeed in transmitting its packet during the truncated sawtooth phase of its epoch, that player joins the next batch.

The players in the current batch take turns humming in a constant fraction of the slots. The first purpose of humming is never to let the channel go silent for more than $O(1)$ slots so that any new inactive player can detect if it entered in the middle of an epoch. This aspect of humming is based on the "busy signal" idea of [13].

The second purpose of humming is to notify new players precisely when the current epoch will end. In the truncated sawtooth the players know exactly how many slots are left, say t, and will announce "t" when it is their turn to hum. Once a new player hears "t" it can sleep for t slots, wake up, and join the next batch. If, however, the current epoch is still in the estimation phase, no player knows when the epoch will end. In this case they encode their current estimate of $\log^* n$ using $O(\log(\log^* n))$ slots (noisy=1, clear=0) so that new players can sleep until truncated sawtooth has begun or a better estimate of $\log^* n$ is known. The listening cost for inactive players to determine when the current epoch will end is $O(\log(\log^* n))$.

Organization. This paper is organized as follows. Section 2 explains how the players can collectively simulate a circuit, proving Theorem 2. The estimation phase is described in sections 3 and 4. The truncated sawtooth is described and analyzed in Section 5. Section 6 describes how humming is introduced into both the estimation and truncated sawtooth phases.

2. Decentralized simulation of circuits. Let C be a constant fan-in circuit with ℓ input bits and c gates $g_1, \ldots, g_c$, listed in some topological order. There are ℓ designated slots that encode the input to the circuit (full=1, empty=0) and n players, who must collectively evaluate C. Every player knows C and the particular topological sort $g_1, \ldots, g_c$, but not n. Our simulation scheme will make use of $\ell + 2c$ time slots. The first c slots are *control* slots, which allow the players to take responsibility for implementing certain gates. We guarantee that at least one player is assigned to each gate. The next ℓ time slots are the input slots, each of which is either empty or full. The last c time slots are *circuit slots* which encode the output bit of each gate.

Control slots. The c control slots enable the players to coordinate among themselves to guarantee that (with probability 1) each gate is simulated by at least one player.

Each player P picks one uniformly random control slot τ_P and broadcasts in that slot. Let τ'_P be the first control slot following τ_P that is full, or $c+1$ if no such slot exists. The player P will be responsible for simulating gates $g_{\tau_P}, \ldots, g_{\tau'_P - 1}$. In order for P to know exactly which gates it is responsible for, P listens to the control slots following τ_P until another player transmits or the control slots end, thereby defining τ'_P.

Some special treatment is needed for the prefix of empty control slots, since there must be at least one player that is responsible for simulating the corresponding gates. To solve this, we have all the players be responsible for simulating these gates, thereby guaranteeing that every gate is simulated.

Input slots. Each input slot is either empty or full (0 or 1), and how it got to be empty or full is not the concern of this simulation. (As we will see in sections 3

and 4, it is the players themselves who make noise in each input slot with some small probability.)

Circuit slots. The ith circuit slot encodes the output of gate g_i. If player P is responsible for simulating g_i then it listens during the slots corresponding to the inputs of g_i (either input slots or previous circuit slots) and sends noise on circuit slot i if and only if the output of g_i is 1. Since all the players simulating g_i will compute the same output bit, they will all either send noise or stay silent.

LEMMA 4. *Each player is responsible for $O(1+\frac{c}{n})$ gates in expectation.*

Proof. Each player P is responsible for g_{τ_P} and a fake gate g_0, and the runs of unclaimed gates immediately following g_0 and g_{τ_P}. A gate g_i is unclaimed if the ith control slot is empty. Each control slot is empty with probability $(1-\frac{1}{c})^n < \exp(-n/c)$. Moreover, conditioned on one slot being empty, the probability that the next is empty is smaller. Thus, the expected number of empty slots in the run following control slot τ_P (or control slot 0) is less than $(1-\exp(-n/c))^{-1}-1$, which is negligible for $n \gg c$ and always $O(\frac{c}{n})$. ∎

Proof of Theorem 2. The circuit is correctly evaluated with probability 1. The only uncertainty is the cost paid by the players. Being responsible for a gate entails listening for $O(1)$ slots (because C was assumed to have bounded fan-in) and sending in the output slot for that gate. The number of gates that a player is responsible for is $O(1+\frac{c}{n})$ in expectation.

Note that if a player wants to learn the output of C, it must also listen to all the circuit slots corresponding to the output gates. ∎

3. A circuit for testing size. Consider the following experiment for determining whether $i \le \log n$ or $i > \log n$. In each of ℓ slots each player sends with probability $1/2^i$. If $i = \log n$ then we expect to see $\ell(1-1/2^i)^n \approx \ell/e$ empty slots. If we see fewer empty slots we conclude $i \le \log n$; if we see more empty slots we conclude $i > \log n$. We are only concerned with the accuracy of this test when $|i - \log n|$ is a sufficiently large constant, e.g., 1. It is clearly likely to give an incorrect answer when i is very close to $\log n$. We use a circuit simulation of $\mathrm{Thr}_{\ell,(1-1/e)\ell}$ (defined below) to let the players determine the result of this test.

LEMMA 5. *For any natural numbers ℓ and t, there exists a $\Theta(\ell)$-gate circuit $\mathrm{Thr}_{\ell,t} : \{0,1\}^\ell \to \{0,1\}$ such that, for any binary string x of length ℓ, $\mathrm{Thr}_{\ell,t}(x) = 1$ if and only if the Hamming weight (number of 1's) of x is at most t.*

Proof. We first construct a circuit to output the Hamming weight of x in binary. Recursively construct two circuits to compute the Hamming weight of the first $\ell/2$ (last $\ell/2$) bits of x, then attach an $O(\log \ell)$-bit adder to compute their sum. The number of gates in this construction is $S(\ell) = 2S(\ell/2) + O(\log \ell)$, which is $O(\ell)$. The circuit $\mathrm{Thr}_{\ell,t}(x)$ first computes the Hamming weight of x, then compares it against the fixed constant t. The comparison only requires $O(\log \ell)$ additional gates. ∎

LEMMA 6. *Define C_i to be the circuit $\mathrm{Thr}_{\ell,(1-1/e)\ell}$, where $\ell = \Omega(i)$ is a parameter, and suppose each of the n players makes noise in each input slot of C_i with probability $1/2^i$. If $i \le \log n - 1$ then the probability that C_i returns 1 is $\exp(-\Omega(\ln n))$. If $i \ge \log n + 1$ then the probability that C_i returns 0 is $\exp(-\Omega(\ell))$.*

Proof. Suppose $\Delta = \log n - i \ge 1$. The probability of a specific input slot being empty is $(1-2^{-i})^n = (1-2^\Delta/n)^n = p < e^{-2^\Delta} \le e^{-2}$. Let X be the number of empty

slots, so $\mathrm{E}(X) = p\ell$. By a Chernoff bound, $\Pr[X \geq \ell/e] = \exp(-\Omega(\ell))$, which is $\exp(-\Omega(\ln n))$ if $\Delta \leq 2$. For $\Delta > 2$ we can obtain stronger bounds by direct analysis without going through Chernoff bounds. The probability of seeing at least ℓ/e empty slots is, by a union bound, at most

$$\begin{aligned}
\binom{\ell}{\ell/e} p^{\ell/e} &< \left(\frac{e\ell}{\ell/e}\right)^{\ell/e} p^{\ell/e} && \text{since } \tbinom{x}{y} < (ex/y)^y \\
&= \exp\left(\frac{2\ell}{e} + \frac{\ell \ln p}{e}\right) \\
&< \exp\left(\frac{(2-2^{\Delta})\ell}{e}\right) && \text{since } \ln p < -2^{\Delta} \\
&< \exp\left(-\Omega(2^{\Delta}\ell)\right) && \text{since } \Delta > 2 \\
&= \exp\left(-\Omega(\ln n)\right) && \text{since } \ell + \Delta = \Omega(\log n).
\end{aligned}$$

Now suppose $\Delta = i - \log n \geq 1$. The probability of an input slot being empty is $(1-2^{-i})^n = (1 - 1/(n2^{\Delta}))^n = p > e^{-1/2^{\Delta}} - o(1) > 1/2$. The expected number of empty slots is $\mathrm{E}(X) = p\ell$. By a Chernoff bound, $\Pr[X \leq \ell/e] = \exp(-\Omega(\ell))$. ∎

4. A protocol for estimating n.

Exponential search. For a sufficiently large constant d, define the quickly growing sequence (X_i) as

$$\begin{aligned}
X_0 &= d, \\
X_i &= 2^{X_{i-1}^{1/3}}.
\end{aligned}$$

LEMMA 7. *Define k to be such that $X_{k-1} \leq n < X_k$. There exists an algorithm in which n players agree on an integer $\hat{i}$ such that the cost per player is $O(\log \hat{i}) = O(\log(\log^* X_{\hat{i}}))$, the probability that $\hat{i} \geq k+2$ (an overestimate) is at most $n/X_{\hat{i}-1}$, and the probability that $\hat{i} \leq k-2$ (an underestimate) is at most $e^{-n/\log^3(n)}$.*

Proof. The n players perform the following experiment to *test a value* i. In a single time slot each player sends independently with probability $1/X_i$. If the slot is full, then we conclude that $i < k$; otherwise we conclude $i \geq k$.

The algorithm uses exponential search on the index i: the players perform repeated doubling on the index i until they find the first value whose slot is empty; then they perform binary search to find the value $\hat{i}$ such that the slot for value $\hat{i}-1$ is full while the slot for value $\hat{i}$ is empty. At this point, the algorithm makes the educated guess that $k = \hat{i}$, which, we show below, is accurate to within 1 with high probability.

Each player listens in every slot, regardless of whether it chooses to send. Moreover, every player knows exactly when the exponential search ends, and the value of $\hat{i}$. Each player listens in $O(\log \hat{i}) = O(\log(\log^* X_{\hat{i}}))$ slots and sends in $O(1)$ slots in expectation.

Suppose we are testing the value i and that $k < i$. If the slot is full then we will erroneously conclude that $\hat{i} \geq i+1 \geq k+2$. This occurs with probability $1 - (1-1/X_i)^n \leq n/X_i$. Thus, if $\hat{i} \geq k+2$ then when testing $\hat{i}-1$ the slot was full, which happens with probability at most $n/X_{\hat{i}-1}$. On the other hand, if we are testing

the value i and $i \leq k-2$ then the probability of seeing an empty slot, and erroneously concluding that $\hat{i} \leq i$, is $(1 - \frac{1}{X_i})^n \leq e^{-n/X_i} = e^{-n/\log^3 X_{i+1}} \leq e^{-n/\log^3 n}$. ∎

Lemma 8 implies that once we have a reasonably good lower bound on n, we can approximate n in $o(n)$ time with expected $O(1)$ channel accesses.

LEMMA 8. *Suppose all players agree on a value i. There exists an algorithm that selects a subset L of the players with the following properties.*

- *At the end of the algorithm every player in L learns that it is in L; all players in L agree on an integer $j \in [\log X_{i-1}, \log X_i]$.*
- *If $L \neq \emptyset$, then $j \in \{\lfloor \log n \rfloor, \lceil \log n \rceil\}$ with probability $1 - 1/\operatorname{poly}(X_i)$. The expected cost per player is $O(1 + X_{i-1}^{2/3}/n)$, and the expected size of L is $O(1 + n/X_{i-1}^{2/3})$.*
- *If $n < X_i/2$, then the probability that $L = \emptyset$ is $1/\operatorname{poly}(X_i)$.*

Proof. Let C'_j be the circuit C_j from Lemma 6 with an additional input bit called the "override bit." The output of C'_j is the OR of the output of C_j and the override bit. The number of input bits to C_j is ℓ, to be specified shortly.

We construct a "supercircuit" SC_i composed of subcircuits

$$C'_{\log X_{i-1}}, C'_{\log X_{i-1}+1}, \ldots, C'_{X_{i-1}^{1/3}}$$

which collectively test whether $n \in [X_{i-1}, X_i]$. The subcircuits are chained such that the output of C'_{j-1} is the override (input) bit to C'_j; the override bit to the first subcircuit is zero. Thus, if at least one of the subcircuits passes the threshold test, the output of the supercircuit is 1.

The goal of supercircuit SC_i is to determine whether $\log n$ lies in the interval $[\log X_{i-1}, X_{i-1}^{1/3}]$ (by searching for a $j \in [\log X_{i-1}, X_{i-1}^{1/3}]$ and declaring $\log n \approx j$). In order to control the probability of errors, we fix ℓ, the number of input bits of each of the subcircuits $\{C_j\}$ of SC_i to be exactly $\Theta(\log(X_i)) = \Theta(X_{i-1}^{1/3})$. Thus, the number of gates in SC_i is $O(X_{i-1}^{2/3})$.

Each input slot of C_j is generated by having all players make noise with probability $1/2^j$. This allows us to apply Lemma 6. Thus, the expected cost for generating input slots for each player is $O(1)$. The supercircuit is simulated using Theorem 2, and so the simulation cost per player is expected $O(1 + X_{i-1}^{2/3}/n)$.

The set L is the set of players that simulate the output gate of the first circuit C'_j that outputs 1. Notice that the players in L necessarily know that they are in L and know the value of j. If L is established, then the expected size of L is the expected number of players to simulate a single gate, which is $O(1 + n/X_{i-1}^{2/3})$.

By Lemma 6, the error probability for any subcircuit C'_j is at most $e^{-\Omega(\log X_i)}$. Thus, if $L \neq \emptyset$ and j is established then the probability that $j \leq \log n - 1$ or $j \geq \log n + 1$ is polynomially small in X_i. Moreover, if $n < X_i/2$, then the only way in which j will not be established is if the last circuit $C'_{X_{i-1}^{1/3}}$ errs, which happens with probability $e^{-\Omega(\log X_i)}$. ∎

Leader election. The members of L know that $\mathrm{E}[|L|] = O(n/X_{i-1}^{2/3})$, and can elect a leader in $O(1)$ time, in expectation, by each sending with probability $1/\mathrm{E}[|L|]$ until precisely one member of L sends without collision. This is the leader. In general we would like to have a leader-election protocol that is more robust to errors in the size of $|L|$. Nakano and Olariu [63] gave sharp bounds on the time complexity of low-error leader election.

LEMMA 9 (see Nakano and Olariu [63]). *There is a protocol for electing a leader among a set L of players ($|L|$ unknown) that takes $O(\log\log|L|)$ time in expectation, and $O(\log\log|L| + \log f^{-1})$ time with probability $1-f$.*

Note that we can upper bound $|L|$ by X_i. By allocating $O(\log X_i)$ time slots for leader election, Lemma 9 ensures that the protocol correctly elects a leader with probability $1 - 1/\operatorname{poly}(X_i)$.

4.1. Estimating n. In this section, we give an efficient protocol enabling n players to estimate n to within a constant factor. We prove that the expected cost to estimate n is only $O(\log(\log^* n))$.

THEOREM 10. *There exists an algorithm for n synchronized players to agree on an integer j such that $j < \log n - 1$ with probability $1/\operatorname{poly}(n)$ and $j > \log n + 1$ with probability $1/\operatorname{poly}(2^j)$. The protocol lasts $O(n^{2/3})$ time with probability $1 - 1/\operatorname{poly}(n)$. In expectation, each player sends $O(1)$ times and listens $O(\log(\log^* n))$ times.*

Proof. Our algorithm for estimating n up to a constant factor makes use of Lemmas 7, 8, and 9. The estimation algorithm has three types of phases.

Exponential-search phase. In the first phase, the players attempt to estimate $\log^* n$ to within an additive constant by using Lemma 7; by the end of the phase, all the players know this estimate. Remarkably, estimating $\log^* n$ is the most expensive part of estimating n.

Estimating-$\log n$ phase. In the second phase, the players attempt to use the estimate of $\log^* n$ to establish a subset L of the players that (1) know that they are in L (and every other player not in L knows that it is not in L), and (2) have the same estimate of $\log n$ which is accurate up to some additive constant. This is established by applying Lemma 8 up to three times, as follows. Let $\hat{i}$ be the value established by the algorithm of Lemma 7. We first simulate $SC_{\hat{i}-1}$, and have all of the players listen to the output of the very last gate, which is 1 iff some subcircuit C'_j outputted 1. If it is 1, some nonempty subset L of players has estimated $\log n$ and the phase ends. Otherwise, we repeat the process by simulating $SC_{\hat{i}}$, and if its last gate outputs 0, then we simulate $SC_{\hat{i}+1}$. If $SC_{\hat{i}+1}$ outputs 0, then we declare failure; all players restart the algorithm from the exponential-search phase.

Leader-election phase. Suppose L is nonempty, and formed in the simulation of $SC_{i^\star}$, where $i^\star \in \{\hat{i}-1, \hat{i}, \hat{i}+1\}$. All members of L agree on an estimate $j \approx \log n$. We apply Lemma 9 for $O(\log X_{i^\star})$ time slots, and elect a leader with probability $1 - 1/\operatorname{poly}(X_{i^\star})$. The leader sends "$n \approx 2^j$" in a predesignated time slot, in which all other players listen. If no leader is elected then this slot is empty, in which case a failure has been detected. All the players restart the algorithm from the exponential-search phase.

Conclusion. At the end of a successful leader-election phase all the players agree on an estimate of $\log n$ which is accurate up to ± 1.

Cost analysis. We now determine the expected cost of each player in the algorithm.

The cost of the exponential-search phase is $O(\log(\log^* X_{\hat{i}}))$, where $\hat{i}$ is the output of the algorithm in Lemma 7. Let k be such that $X_{k-1} \le n < X_k$, so $k = \Theta(\log^* n)$. Since the probability of obtaining an overestimate is negligible, the expected cost of the exponential search is $O(\log(\log^* n))$.

The cost of the estimating-$\log n$ phase is a bit more involved. The phase first simulates $SC_{\hat{i}-1}$ using Lemma 8, and so the expected cost of each player is $O(X_{\hat{i}-2}^{2/3}/n)$.

If $\hat{i} \leq k+1$ this cost is $O(1 + X_{\hat{k}-1}^{2/3}/n) \leq O(1 + n^{2/3}/n) = O(1)$. In general the probability of having an erroneous $\hat{i} \geq k+2$ is at most $n/X_{\hat{i}-1}$, so the expected cost per player to simulate $SC_{\hat{i}-1}$ in these circumstances is at most

$$\sum_{\hat{i} \geq k+2} (n/X_{\hat{i}-1})(1 + X_{\hat{i}-2}^{2/3}/n) = O(1).$$

The probability of having an erroneous $\hat{i} \leq k-2$ is $\exp(-n/\log^3 n)$, and the expected cost for restarting the algorithm is negligible in this case.

By Lemma 8, either $n \geq X_{\hat{i}-1}/2$ or $\Pr(SC_{\hat{i}-1} \text{ outputs } 0) = 1/\operatorname{poly}(X_{\hat{i}-1})$. In either case, the expected cost of simulating the next supercircuit $SC_{\hat{i}}$ is

$$\Pr(SC_{\hat{i}-1} \text{ outputs } 0) \cdot O(1 + X_{\hat{i}-1}^{2/3}/n) = O(1 + n^{-1/3}) = O(1).$$

By the same reasoning, the expected cost of simulating $SC_{\hat{i}+1}$ is

$$\Pr(\text{Both } SC_{\hat{i}-1} \text{ and } SC_{\hat{i}} \text{ output } 0) \cdot O(1 + X_{\hat{i}}^{2/3}/n) = O(1).$$

Let C'_j be the first subcircuit to output 1 in $SC_{i^\star}$ for some $i^\star \in \{\hat{i}-1, \hat{i}, \hat{i}+1\}$. By Lemma 6, $j \in \{\lfloor \log n \rfloor, \lceil \log n \rceil\}$ with probability $1 - 1/\operatorname{poly}(\max\{n, X_{i^\star}\})$. The set L of players simulating the output of C'_j know that $|L| = O(X_{i^\star})$ with probability $1 - 1/\operatorname{poly}(X_{i^\star})$. By Lemma 9, L correctly elects a leader in $O(\log X_{i^\star})$ slots with probability $1 - 1/\operatorname{poly}(X_{i^\star})$. The expected cost per player P to participate in leader election is $\Pr(P \in L) \cdot O(\log X_{i^\star}) = O(1/X_{i^\star-1}^{2/3}) \cdot O(X_{i^\star-1}^{1/3}) = o(1)$. ∎

5. The truncated sawtooth. The sawtooth protocol of Bender et al. [11] achieves constant throughput for a single batch of n players without knowing n, but at a cost of $\Omega(\log n)$ transmission attempts per player because of the exponential search for n. However, in our protocol since the n players already have an estimate $\tilde{n}$ of n, they can jump directly to the correct iteration of the sawtooth protocol and send all packets in $O(\tilde{n})$ slots with high probability and using an expected $O(1)$ channel accesses.

In this section we assume $\tilde{n} \geq n$. Recall that by the results in section 4, with high probability $\tilde{n} \geq n/2$. Therefore, the inequality $\tilde{n} \geq n$ can be guaranteed simply by multiplying the estimate by a factor of 2; this is implicitly assumed throughout.

The truncated sawtooth protocol. We partition $O(\tilde{n})$ slots into $\log\log\tilde{n} + O(1)$ windows. The 0th window has length precisely $2\tilde{n}$ and, in general, the ith window has length $2\tilde{n}/\alpha^i$ for some $\alpha > 1$.[4] The total length of all windows is less than $2\tilde{n}(\alpha-1)^{-1} = O(\tilde{n})$. At the ith window, each player that has yet to successfully transmit its packet broadcasts in a slot chosen uniformly at random from the window. If there is no collision, the player detects this and refrains from participating in the remaining windows.

Analysis. We prove that with high probability the number of active players at the beginning of window i is at most $n/f(i)$, where f is a function to be determined. Suppose the claim holds at the beginning of window i. Since at most $n/f(i)$ of the $2\tilde{n}/\alpha^i$ slots are full, the probability that a particular player fails to send its packet is at most $\frac{n/f(i)-1}{2\tilde{n}/\alpha^i} < \frac{\alpha^i}{2f(i)}$, *independently of the choices of all other players*. Call a window

[4] The sawtooth algorithm of [11] fixes the decay factor α to be 2. We find it helpful in the analysis to keep α a named parameter.

successful if the number of failures is at most α times its expectation, that is, at most $\frac{n\alpha^{i+1}}{2f^2(i)}$. By a Chernoff bound, $\Pr(\text{window } i \text{ is successful}) < \exp(-\Omega(\frac{(\alpha-1)^2\cdot\alpha^i n}{2f^2(i)}))$.

Given this definition of success we set $f(i+1) = 2f^2(i)/\alpha^{i+1}$. One can prove by induction that $f(i) = 2^{2^i-1}/\alpha^{2^{i+1}-i-2}$. Note that for small α, say $\alpha = 1.1$, $f(\log\log n + O(1)) > n$ so the first $\log\log n + O(1)$ windows drastically reduce the number of active players. However, the probability of success (as defined above) becomes very low when $f^2(i)$ is close to n. Once $f(i)$ is polynomial, say at least $n^{1/5}$, the current window size is still rather large: $2\tilde{n}/\alpha^{\log\log n+O(1)} \gg n/\log n$. Once the disparity between the number of active players and window size is this large, it is easy to see that all players successfully send their packets in the next $O(1)$ windows, with probability $1 - 1/\operatorname{poly}(n)$.

LEMMA 11. *Suppose a batch of n players run the truncated sawtooth protocol and all agree on an upper bound $\tilde{n} \geq n$. The protocol takes $O(\tilde{n})$ time slots and with probability $1-1/\operatorname{poly}(n)$ all n players send their packets. The sending cost per player is $O(1)$ in expectation and $\log\log\tilde{n} + O(1)$ in the worst case.*

6. Humming. Players arrive at arbitrary times and must be organized into batches that are *synchronized*, that is, they all agree on a time 0. Once a batch of n players is formed, the players obtain an upper bound $\tilde{n} \geq n$ and then run the truncated sawtooth protocol in $O(\tilde{n})$ slots.

Consider the point of view of a new player entering the system. If there is no active batch running, then the player should be able to detect this and start a new batch, possibly with other players who entered at the same time. Otherwise, if there is an active batch the new player should determine *precisely* how many slots it has left, say t, so it can join a new batch beginning in $t+1$ slots. One difficulty is that players within the active batch only know how many slots are left once they compute $\tilde{n}$ and enter the sawtooth. Thus, we need a mechanism with the following properties:

- New players can determine, by listening to $O(\log(\log^* n))$ slots, whether there is an active batch and, if so, precisely how many slots remain.
- The sending cost per player in the current batch should be $O(1)$ in expectation.

To achieve these ends we have to ensure that an active batch never lets the channel go silent for more than a constant number of slots at a time, because this would confuse a new player into thinking no batch is active. Thus, the n players in the current batch collectively *hum* in at least one in every constant number of slots to keep new players from joining prematurely.

Humming in the estimation phase. We imagine a *baton* being passed between groups of players; at time 0 in the estimation phase all players hold the baton. The baton holders are responsible for both humming and encoding the current estimate ρ of $\log^* n$. Until the estimation phase finishes its binary search to find an i for which $n \in [X_{i-1}, X_{i+1}]$, ρ is 0. When the estimation phase is simulating circuits that test for $n \in [X_{i-1}, X_i)$, $\rho = i$.

The time slots are partitioned into *chunks* of seven consecutive slots, only one of which is dedicated to executing the estimation phase proper. The remaining slots implement baton passing, humming, and encoding ρ.

(0) *baton:* Any player wishing to take the baton broadcasts noise in this slot. All other current baton holders listen in this slot to determine if they still hold the baton.

(1,2) *humming:* Any player holding the baton hums for two consecutive slots.
(3) *silence:* This slot is always empty.
(4) *rho:* Whoever holds the baton sends the next bit in the encoding of ρ: noise = 1 and silence = 0.
(5) *silence:* This slot is always empty.
(6) *data:* This slot is used to implement the estimation phase algorithm.

Observe that there are never six consecutive silent slots, and that any new player can deduce the current time slot modulo 7 by listening to $O(1)$ consecutive slots: it listens for two, three, or four consecutive noisy slots (which can only be slots (1,2) or (0,1,2) or (6,0,1,2)) followed by a silent slot, which must be slot (3).

If the binary representation of ρ is $b_1 b_2 \cdots b_\ell$, the baton holders repeatedly encode ρ in the rho slots as $110 b_1 0 b_2 0 \cdots b_\ell 0$. A listener can decode ρ by listening to $4\ell + O(1) = O(\log \rho)$ chunks: at most $2\ell + O(1)$ chunks to hear two consecutive 1's and $2\ell + O(1)$ more to determine ρ.

At all times the algorithm that estimates n has computed a *likely lower bound* $\bar{n}$ on n.[5] Each player (even one that holds the baton) makes noise in the baton slot with probability $1/\bar{n}$, thereby taking the baton from whatever group that holds it. Thus, any player that currently holds the baton is relieved of duty with probability $(1-1/\bar{n})(1-(1-1/\bar{n})^{n-1})$, which is $\Theta(1)$ for $n \geq \bar{n} > 1$. In the first chunk, all players hold the baton. Since the estimation phase lasts for $O(n^{2/3})$ slots in expectation, each player holds the baton for $O(1)$ chunks in expectation.

Humming in the truncated sawtooth. In this phase all nodes in the active batch have agreed on an estimate $\tilde{n}$, which is $\Theta(n)$ with high probability, and know precisely when the truncated sawtooth phase ends. They need to communicate this information to new players joining the system. Slots are partitioned into chunks of three. The players maintain the invariant that *exactly* one player holds the baton at any given time. The initial baton holder is the elected leader who announced "$\tilde{n}$" at the end of the estimation phase.

(0) *baton:* Every nonbaton holder that wishes to take the baton broadcasts a message in this slot. If exactly one player broadcasts, he takes the baton; if zero or more than one players broadcast, the current baton holder retains the baton.
(1) *humming:* The (unique) baton holder sends "t" if there are t slots remaining in the truncated sawtooth phase.
(2) *data:* This slot is used to implement the truncated sawtooth algorithm.

Each non-baton-holder attempts to take the baton in slot (0) with probability $1/\tilde{n}$. Thus, the current baton holder is relieved of duty with probability $(n-1)(1/\tilde{n})(1-1/\tilde{n})^{n-2}$, which is $\Theta(1)$ for $n > 1$ and $n = O(\tilde{n})$. Since there are $O(\tilde{n})$ chunks each player attempts to take the baton $O(1)$ times in expectation. By symmetry each player holds the baton for $O(\tilde{n}/n)$ chunks in expectation.

Consider how the humming protocol allows new players to cheaply determine when the current batch will finish. If a player joins during the truncated sawtooth, it listens for at most three slots and learns t: the exact number of remaining slots. If a player joins during the estimation phase it listens until it decodes an estimate $\rho > 0$, indicating that the algorithm is simulating circuits testing for $n \in [X_{\rho-1}, X_\rho)$. Let $L_\rho = O(\log^2 X_\rho)$ be the length of the simulation. In general, whenever the player decodes ρ it sleeps for L_ρ slots and begins listening again. If the estimation phase is successful then the new player may listen for $O(\log \rho) = O(\log(\log^* n))$ slots in

[5] In the binary search for $\log^* n \pm 1$ this lower bound can be updated in every time slot. When simulating the circuits that test for $n \in [X_{i-1}, X_i)$ we use X_{i-1} as a lower bound $\bar{n}$ on n.

four intervals: it may decode a ρ, sleep for L_ρ steps, decode $\rho+1$, sleep for $L_{\rho+1}$ steps, decode $\rho+2$, sleep for $L_{\rho+2}$ steps, then listen in the truncated sawtooth and determine exactly how many slots remain.

Pulling the pieces together, we can now give a proof of Theorem 1.

Proof of Theorem 1. Define n_i to be the number of players that participate in the ith epoch. By Theorem 10 and Lemma 11, with high probability in n_i, the estimation and truncated sawtooth phases take $O(n_i^{2/3}+n_i)=O(n_i)$ slots and transmit all packets. Moreover, each player in the batch listens in $O(\log(\log^* n_i))$ slots and sends in $O(1)$ slots, in expectation. Under these circumstances the throughput is constant.

The throughput can suffer if n_i is underestimated (few packets will be sent in this epoch due to high contention) or overestimated (the epoch will likely be successful, but take too long) or if n_i is correctly estimated but collisions prevent a large number of packets from being transmitted. By Theorem 10, the probability that n_i is underestimated is $1/\operatorname{poly}(n_i)$ and the probability that it is overestimated as $\tilde{n}_i > 2n_i$ is $1/\operatorname{poly}(\tilde{n}_i)$. By Lemma 11, when n_i is correctly estimated, the probability that not all packets are sent is $1/\operatorname{poly}(n_i)$. Call a slot *failed* if it is in an epoch that failed due to one of these three causes. Since there are $O(\tilde{n}_i)$ slots in the epoch, the expected number of failed slots per epoch is a tiny $1/\operatorname{poly}(n_i)$, implying that the expected throughput is constant.

A player injected during epoch i will listen to $O(\log(\log^* n_i))$ slots before joining epoch $i+1$, then listen for $O(\log(\log^* n_{i+1}))$ slots in epoch $i+1$, in expectation. With very small probability it will fail to send its packet and continue on to batch $i+2$, and so on. In general, its expected cost in epoch $i' \geq i+2$ is at most $O(\log(\log^* n_{i'})) \cdot \prod_{i''\in[i+1,i'-1]} \frac{1}{\operatorname{poly}(n_{i''})}$. In total the expected listening cost is $O(\log(\log^* N))$, where $N=\sum_j n_j$. The sending cost per player is $O(1)$ in expectation in both the estimation and truncated sawtooth phases. ∎

7. Discussion and conclusion. In this section, we discuss potential applications of our result, as well as highlight some issues for future work that might need to be addressed in order to develop a deployable protocol.

7.1. Application in low-power wireless networks. The small number of channel accesses provided by our algorithm is perfectly suited to low-power wireless networks. In this setting, typically only a single device may send data at any given time, and accessing the shared channel (either for sending or listening) incurs an energy cost which is significant for battery-powered devices. For example, in the wireless sensor network setting, the *send* (at a transmit power of 0 dBm) and *listen* costs for the popular Telos motes [67] are 38 mW and 35 mW, respectively, and these are the dominant operational costs for an active device. Similar costs exist for the older, well known MICA family of devices [36].

A topical application domain is the emerging *internet of things* (IoT), where there is ubiquitous connectivity via all manner of devices for use in personal-wearable technology, city maintenance, smart homes, and many other settings. There were more than 6.4 billion IoT devices (*excluding* phones and laptops) in use at the end of 2016 [74], and 20 to 30 billion are predicted by 2020 [38, 58], with the majority due to consumer use. An energy-efficient mechanism for contention resolution will continue to be an enduring challenge for these devices.

This growth in IoT is likely to coincide with increasingly dense networks. In particular, Bell Labs envisions dense wireless networks as the only way in which to handle

predicted increases in wireless network traffic [80]. Therefore, while $O(\log(\log^* N))$ channel access attempts may not greatly improve the performance in current networks, such benefits may be important to future single-hop networks of greatly increased size and density.

Finally, the ability to handle arbitrary packet arrivals may be valuable in such future networks. As discussed in section 1.3, prior results consider Poisson arrivals. However, it is reasonable to assume that traffic will be bursty (as seen in practice [78, 82]) or even *adversarially* chosen.

7.2. Practical considerations. Our contention-resolution algorithm is intricate and serves as a "proof-of-concept." Significant work would be required to develop a practical protocol and such an effort is beyond the scope of this current work. However, we highlight two significant practical considerations.

First, our algorithm requires that devices be tightly synchronized with respect to when each slot begins and ends. For example, disagreement on even a single slot could cause the humming (section 6) to fail. While synchronization is a challenge in wireless networks, there is reason to be hopeful. In wireless sensor networks, global scheduling has been demonstrated by experimental work in [55], and synchronization has been shown even under adversarial circumstances in [40].

Second, nonmalicious faults in either the devices sending packets or on the channel can lead to poor performance. If devices can fail, then a popular technique for increasing fault tolerance is *state machine replication* [75] where actions are performed in concert in order to avoid dependence on a single device. This redundancy will increase the communication overhead, but for a small amount of replication, the asymptotic guarantees should remain unchanged and the impact from such errors can be reduced.

Faults on the channel may be harder to address since wireless channels are notoriously error prone. For example, see [25, 4]. While techniques for detecting and correcting errors in data transmission are available, a more challenging issue is unintended noise; for example, this may cause a gross overestimate of n (section 4) or corrupt our humming mechanism (section 6). A simple method for mitigating this problem is to again use redundancy by having each slot repeated for, say, a constant number of instances. If the majority of the redundant slots are silent, then the corresponding slot in the original algorithm is considered to be silent.

7.3. Future work. One flaw in our algorithm is that the number of channel accesses is only $O(\log(\log^* n))$ *in expectation*, not with high probability. The high probability guarantee for the worst-case cost of a player in our algorithm is $O(\log n)$, and this cost can be realized at two places in the protocol. Leader election can potentially cost this much (Lemma 9), and in the truncated sawtooth phase, a player may be responsible for humming $\Omega(\log n)$ slots. A realistic goal is to reduce the expected maximum cost of any player to $O(\log \log n)$ and keep the expected cost $O(\log(\log^* n))$.

Extending our work to multihop networks is an interesting open problem. For example, in large wireless networks, not all devices may be able to communicate directly with all other devices due to a limited broadcast range.

Another avenue to explore is the impact of more realistic physical-layer models. For example, the SINR model [7, 62, 39] is less strict about failure in the event of simultaneous transmissions. An alternative that has received attention is the affectance model [50]. It would be of interest to learn the asymptotic impact of these models on our approach.

REFERENCES

[1] *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks – Specific Requirements - Part* 11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012), (2016).

[2] L. Anantharamu, B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki, *Medium access control for adversarial channels with jamming*, in Proceedings of the 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Springer, Berlin, 2011, pp. 89–100.

[3] L. Anantharamu, B. S. Chlebus, and M. A. Rokicki, *Adversarial multiple access channel with individual injection rates*, in Proceedings of the 13th International Conference on Principles of Distributed Systems (OPODIS), Springer, Heidelberg, 2009, pp. 174–188.

[4] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, *Performance measurements of motes sensor networks*, in Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '04, ACM, New York, 2004, pp. 174–181.

[5] W. C. Anderton and M. Young, *Is our model for contention resolution wrong?: Confronting the cost of collisions*, in Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '17, ACM, New York, 2017, pp. 183–194.

[6] A. F. Anta, M. A. Mosteiro, and J. R. Muñoz, *Unbounded contention resolution in multiple-access channels*, Algorithmica, 67 (2013), pp. 295–314.

[7] C. Avin, Y. Emek, E. Kantor, Z. Lotker, D. Peleg, and L. Roditty, *SINR diagrams: Towards algorithmically usable SINR models of wireless networks*, in Proceedings of the 28th ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2009, pp. 200–209.

[8] B. Awerbuch, A. Richa, and C. Scheideler, *A jamming-resistant* MAC *protocol for single-hop wireless networks*, in Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2008, pp. 45–54.

[9] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, *Balanced allocations*, SIAM J. Comput., 29 (1999), pp. 180–200.

[10] R. Bar-Yehuda, O. Goldreich, and A. Itai, *On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization*, J. Comput. System Sci., 45 (1992), pp. 104–126.

[11] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, and C. E. Leiserson, *Adversarial contention resolution for simple channels*, in Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, ACM, New York, 2005, pp. 325–332.

[12] M. A. Bender, J. T. Fineman, and S. Gilbert, *Contention resolution with heterogeneous job sizes*, in Proceedings of the 14th Annual European Symposium on Algorithms (ESA), Springer, Berlin, 2006, pp. 112–123.

[13] M. A. Bender, J. T. Fineman, S. Gilbert, and M. Young, *How to scale exponential backoff: Constant throughput, polylog access attempts, and robustness*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2016, pp. 636–654.

[14] M. A. Bender, T. Kopelowitz, S. Pettie, and M. Young, *Contention resolution with log-logstar channel accesses*, in Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing, STOC '16, ACM, New York, 2016, pp. 499–508.

[15] P. Berenbrink, A. Czumaj, M. Englert, T. Friedetzky, and L. Nagel, *Multiple-choice balanced allocation in (almost) parallel*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM), Springer, Berlin, 2012, pp. 411–422.

[16] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking, *Balanced allocations: The heavily loaded case*, SIAM J. Comput., 35 (2006), pp. 1350–1385.

[17] P. Berenbrink, K. Khodamoradi, T. Sauerwald, and A. Stauffer, *Balls-into-bins with nearly optimal load distribution*, in Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, 2013, pp. 326–335.

[18] D. J. Bernstein, *qmail*, http://cr.yp.to/qmail.html (1998).

[19] G. Bianchi, *Performance analysis of the IEEE* 802.11 *distributed coordination function*, IEEE J. Sel. Areas Commun., 18 (2006), pp. 535–547.

[20] G. Bianchi and I. Tinnirello, *Kalman filter estimation of the number of competing terminals in an IEEE* 802.11 *network*, in Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Vol. 2, IEEE, Piscataway, NJ, 2003, pp. 844–852.

[21] P. Brandes, M. Kardas, M. Klonowski, D. Pajak, and R. Wattenhofer, *Approximating the size of a radio network in beeping model*, in Proceedings of the 23rd International Colloquium Structural Information and Communication Complexity (SIROCCO), Springer, Cham, Switzerland, 2016, pp. 358–373.

[22] F. Cali, M. Conti, and E. Gregori, *Dynamic tuning of the IEEE* 802.11 *protocol to achieve a theoretical throughput limit*, IEEE/ACM Trans. Netw., 8 (2000), pp. 785–799.

[23] F. Cali, M. Conti, and E. Gregori, *IEEE* 802.11 *protocol: Design and performance evaluation of an adaptive backoff mechanism*, IEEE J. Sel. Areas Commun., 18 (2000), pp. 1774–1786.

[24] A. Casteigts, Y. Métivier, J. M. Robson, and A. Zemmari, *Counting in One-Hop Beeping Networks*, preprints, arXiv:1605.09516, 2016.

[25] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, *Statistical model of lossy links in wireless sensor networks*, in Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN), IEEE, Piscataway, NJ, 2005, pp. 81–88.

[26] Y. Chang, T. Kopelowitz, S. Pettie, R. Wang, and W. Zhan, *Exponential separations in the energy complexity of leader election*, in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, 2017, ACM, New York, 2017, pp. 771–783.

[27] B. S. Chlebus, G. De Marco, and D. R. Kowalski, *Scalable wake-up of multi-channel single-hop radio networks*, Theoret. Comput. Sci., 615 (2016), pp. 23–44.

[28] B. S. Chlebus, L. Gasieniec, D. R. Kowalski, and T. Radzik, *On the wake-up problem in radio networks*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP), Springer, Berlin, 2005, pp. 347–359.

[29] B. S. Chlebus and D. R. Kowalski, *A better wake-up in radio networks*, in Proceedings of 23rd ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2004, pp. 266–274.

[30] B. S. Chlebus and D. R. Kowalski, *Randomization helps to perform independent tasks reliably*, Random Structures Algorithms, 24 (2004), pp. 11–41.

[31] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki, *Adversarial queuing on the multiple access channel*, ACM Trans. Algorithms, 8 (2012), 5.

[32] M. Chrobak, L. Gasieniec, and D. R. Kowalski, *The wake-up problem in multihop radio networks*, SIAM J. Comput., 36 (2007), pp. 1453–1471.

[33] R. Cole, A. M. Frieze, B. M. Maggs, M. Mitzenmacher, A. W. Richa, R. K. Sitaraman, and E. Upfal, *On balls and bins with deletions*, in Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM), Springer, Berlin, 1998, pp. 145–158.

[34] A. Cornejo and F. Kuhn, *Deploying wireless networks with beeps*, in Proceedings of the 24th International Symposium on Distributed Computing (DISC), Springer, Berlin, 2010, pp. 148–162.

[35] B. Costales and E. Allman, *Sendmail*, 3rd ed., O'Reilly, Sebastopol, CA, 2002.

[36] Crossbow, MICAz Wireless Measurement System. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

[37] G. De Marco and G. Stachowiak, *Asynchronous shared channel*, in Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC '17, ACM, New York, 2017, pp. 391–400.

[38] Ericsson, *Ericsson Mobility Report*, https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf (2016).

[39] J. T. Fineman, S. Gilbert, F. Kuhn, and C. Newport, *Contention resolution on a fading channel*, in Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2016, pp. 155–164.

[40] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava, *Secure time synchronization in sensor networks*, ACM Trans. Inform. System Security, 11 (2008), pp. 1–35.

[41] M. Geréb-Graus and T. Tsantilas, *Efficient optical communication in parallel computers*, in Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), ACM, New York, 1992, pp. 41–48.

[42] S. Gilbert and C. Newport, *The computational power of beeps*, in Proceedings of the International Symposium on Distributed Computing (DISC), Springer, Berlin, 2015, pp. 31–46.

[43] L. A. Goldberg and P. D. MacKenzie, *Analysis of practical backoff protocols for contention resolution with multiple servers*, J. Comput. System Sci., 58 (1999), pp. 232–258, https://doi.org/10.1006/jcss.1998.1590.

[44] L. A. Goldberg, P. D. Mackenzie, M. Paterson, and A. Srinivasan, *Contention resolution with constant expected delay*, J. ACM, 47 (2000), pp. 1048–1096.

[45] J. Goodman, A. G. Greenberg, N. Madras, and P. March, *Stability of binary exponential backoff*, J. ACM, 35 (1988), pp. 579–602.

[46] Google, Google Cloud Messaging: Overview, http://developer.android.com/google/gcm/adv.html#retry (2014).

[47] A. G. Greenberg, P. Flajolet, and R. E. Ladner, *Estimating the multiplicities of conflicts to speed their resolution in multiple access channels*, J. ACM, 34 (1987), pp. 289–325.

[48] A. G. Greenberg and S. Winograd, *A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels*, JACM, 32 (1985), pp. 589–596.

[49] R. I. Greenberg and C. E. Leiserson, *Randomized routing on fat-trees*, in Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Angeles, 1985, pp. 241–249.

[50] M. M. Halldórsson and R. Wattenhofer, *Wireless communication is in APX*, in 36th International Colloquium on Automata, Languages and Programming (ICALP), Springer, Berlin, 2009, pp. 525–536.

[51] J. Håstad, T. Leighton, and B. Rogoff, *Analysis of backoff protocols for multiple access channels*, SIAM J. Comput., 25 (1996), pp. 740–774.

[52] M. Herlihy and J. E. B. Moss, *Transactional memory: Architectural support for lock-free data structures*, in Proceedings of the 20th International Conference on Computer Architecture, IEEE Computer Society, Los Alamitos, CA, 1993, pp. 289–300, https://dl.acm.org/citation.cfm?id=165164.

[53] V. Jacobson, *Congestion avoidance and control*, SIGCOMM Comput. Commun. Rev., 18 (1988), pp. 314–329.

[54] T. Jurdzinski and G. Stachowiak, *The cost of synchronizing multiple-access channels*, in Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2015, pp. 421–430.

[55] Y. Li, W. Ye, and J. Heidemann, *Energy and latency control in low duty cycle MAC protocols*, in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), IEEE, Piscataway, NJ, 2005, pp. 676–682.

[56] G. D. Marco and D. R. Kowalski, *Fast nonadaptive deterministic algorithm for conflict resolution in a dynamic multiple-access channel*, SIAM J. Comput., 44 (2015), pp. 868–888.

[57] G. D. Marco and D. R. Kowalski, *Contention resolution in a non-synchronized multiple access channel*, Theoret. Comput. Sci., 689 (2017), pp. 1–13.

[58] IHS Markit, *IoT Trend Watch* 2017, https://www.ihs.com/info/0117/IoT-Trend-Watch-2017.html (2017).

[59] R. M. Metcalfe and D. R. Boggs, *Ethernet: Distributed packet switching for local computer networks*, Commun. ACM, 19 (1976), pp. 395–404.

[60] M. Mitzenmacher, *The power of two choices in randomized load balancing*, IEEE Trans. Parallel Distrib. Syst., 12 (2001), pp. 1094–1104.

[61] A. Mondal and A. Kuzmanovic, *Removing exponential backoff from TCP*, SIGCOMM Comput. Commun. Rev., 38 (2008), pp. 17–28.

[62] T. Moscibroda, *The worst-case capacity of wireless sensor networks*, in Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN), IEEE, Piscataway, NJ, 2007, pp. 1–10.

[63] K. Nakano and S. Olariu, *Uniform leader election protocols for radio networks*, IEEE Trans. Parallel Distrib. Syst., 13 (2002), pp. 516–526, https://doi.org/10.1109/TPDS.2002.1003864.

[64] C. Newport, *Radio network lower bounds made easy*, in Proceedings of the 28th International Symposium on Distributed Computing (DISC), Springer, Berlin, 2014, pp. 258–272, https://doi.org/10.1007/978-3-662-45174-8_18.

[65] A. Ogierman, A. Richa, C. Scheideler, S. Schmid, and J. Zhang, *Sade: competitive MAC under adversarial SINR*, Distrib. Comput., 31 (2018), pp. 241–254.

[66] G. A. Platform, *Google Documents List API Version* 3.0: *Implementing Exponential Backoff*. https://cloud.google.com/iot/docs/how-tos/exponential-backoff, 2011.

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

[68] P. Raghavan and E. Upfal, *Stochastic contention resolution with short delays*, SIAM J. Comput., 28 (1998), pp. 709–719.
[69] R. Rajwar and J. R. Goodman, *Speculative lock elision: Enabling highly concurrent multithreaded execution*, in Proccedings of the 34th Annual International Symposium on Microarchitecture, Austin, Texas, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 294–305, http://www.cs.wisc.edu/~rajwar/papers/micro01.pdf.
[70] A. Richa, C. Scheideler, S. Schmid, and J. Zhang, *A jamming-resistant MAC protocol for multi-hop wireless networks*, in Proceedings of the International Symposium on Distributed Computing (DISC), Springer, Berlin, 2010, pp. 179–193.
[71] A. Richa, C. Scheideler, S. Schmid, and J. Zhang, *Competitive and fair medium access despite reactive jamming*, in Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS), IEEE, Piscataway, NJ, 2011, pp. 507–516.
[72] A. Richa, C. Scheideler, S. Schmid, and J. Zhang, *Competitive and fair throughput for co-existing networks under adversarial interference*, in Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2012, pp. 291–300.
[73] A. W. Richa, M. Mitzenmacher, and R. Sitaraman, *The power of two random choices: A survey of techniques and results*, Comb. Optimization, 9 (2001), pp. 255–304.
[74] G. Rob van der Meulen, *Gartner Says* 8.4 *Billion Connected "Things" Will Be in Use in* 2017*, Up* 31 *Percent from* 2016, http://www.gartner.com/newsroom/id/3598917 (2017).
[75] F. B. Schneider, *Implementing fault-tolerant services using the state machine approach: A tutorial*, ACM Comput. Surv., 22(4) (1990), pp. 299–319.
[76] A. W. Services, *Error Retries and Exponential Backoff in AWS*, http://docs.aws.amazon.com/general/latest/gr/api-retries.html, 2012.
[77] N.-O. Song, B.-J. Kwak, and L. E. Miller, *On the stability of exponential backoff*, J. Res. Natl. Inst. Standards Technol., 108 (2003), pp. 289–297.
[78] S. Teymori and W. Zhuang, *Queue analysis for wireless packet data traffic*, in International Conference on Research in Networking, Springer, Berlin, 2005, pp. 217–227.
[79] B. Vöcking, *How asymmetry helps load balancing*, J. ACM, 50 (2003), pp. 568–589.
[80] M. K. Weldon, *The Future X Network: A Bell Labs Perspective*, 1st ed., CRC Press, Boca Raton, FL, 2015.
[81] D. E. Willard, *Log-logarithmic selection resolution protocols in a multiple access channel*, SIAM J. Comput., 15 (1986), pp. 468–477.
[82] J. Yu and A. P. Petropulu, *Study of the effect of the wireless gateway on incoming self-similar traffic*, IEEE Trans. Signal Process., 54 (2006), pp. 3741–3758.