Contention Resolution with Heterogeneous Job Sizes

Michael A. Bender[1], Jeremy T. Fineman[2], and Seth Gilbert[2]

[1] Dept. of Computer Science, Stony Brook University, NY 11794-4400, USA.
bender@cs.sunysb.edu
[2] CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
{jfineman,sethg}@mit.edu

Abstract. We study the problem of contention resolution for different-sized jobs on a simple channel. When a job makes a run attempt, it learns only whether the attempt succeeded or failed. We first analyze binary exponential backoff, and show that it achieves a makespan of $V2^{\Theta(\sqrt{\log n})}$ with high probability, where V is the total work of all n contending jobs. This bound is significantly larger than when jobs are constant sized. A variant of exponential backoff, however, achieves makespan $O(V \log V)$ with high probability. Finally, we introduce a new protocol, size-hashed backoff, specifically designed for jobs of multiple sizes that achieves makespan $O(V \log^3 \log V)$. The error probability of the first two bounds is polynomially small in n and the latter is polynomially small in $\log V$.

1 Introduction

Randomized backoff is a common mechanism for reducing contention on a shared resource. Processes/jobs make competing attempts to access the resource, but only one can gain control of the resource at a time. If an access attempt fails due to contention, then that process waits for a random amount of time before trying again. On subsequent failed attempts, the waiting time increases, thereby reducing the probability of a collision and increasing the chance of successful resource acquisition.

Backoff is used in many contexts, for example, network access (e.g., an Ethernet bus [1]), wireless communication [2], transactional memory [3], and speculative-lock elision [4]. In these and other applications of randomized backoff, the lengths of jobs fluctuate substantially. Most theoretical analyses, however, assume unit-length jobs. In a transactional shared-memory system, for example, jobs (transactions) can vary by four to five orders of magnitude [5]. In a wireless network, jobs (packet transmissions) can vary by over three orders of magnitude. The job length is proportional to both transmission length (in bits) and

This research was supported in part by the Singapore-MIT Alliance, NSF Grants CCR-0208670, ITR-0121277, CNS-0305606, OCI-0324974, and by USAF/AFRL Award #FA9550-04-1-0121.

the transmission speed; the speed of the transmitters alone varies considerably (e.g., from roughly 10Kb/s to 10Mb/s).

This paper gives the first theoretical analysis of randomized backoff when jobs have variable sizes. We analyze a system consisting of jobs $1, \ldots, n$. Job i has size $t_i \geq 1$, which indicates that i must run for t_i consecutive units of time in order to complete. We define the ***volume*** of the jobs as $V = \sum_{i=1}^{n} t_i$. Each job knows its own size, but does not know any other job size or the number of other jobs. For simplicity, we assume that t_i is integral and that time is divided into unit-sized time slots, but our analyses extend to the case of nonintegral sizes.

The jobs are competing for access to a ***simple channel*** and have no other means of communication. Whenever a job of size t_i makes a run attempt, it must execute for the full t_i consecutive timeslots. If a job's run is uncontested, then the job completes successfully. If multiple jobs make overlapping run attempts, then all attempts fail and the jobs must retry. A job i learns whether its run attempt is successful only *after* the full t_i time slots, not instantly when the collision occurs. A job gains information only by making run attempts—there is no "listening" on the channel. No other information (e.g., the number of jobs that made attempts in a time slot) is available to the job. (These assumptions are roughly the worst case, in terms of information learned when a collision occurs.)

In this paper, we consider the ***batch problem*** (also called the ***control-tower problem*** [6] or ***shopping-cart problem*** [7, 8]), where all jobs arrive at time 0. We analyze the worst-case ***makespan*** of the protocols, which is the maximum completion time among all the jobs.

This paper discusses ***windowed backoff protocols*** in which time is divided into a sequence of windows $\langle W_1, W_2, W_3, \ldots \rangle$. A job makes at most one run attempt in any window. Notice that a job can make a run attempt only if the window is larger than the job size. Even if a job does fit in a window, it may choose not to make a run attempt. If the job does choose to execute, it randomly chooses a position in the window such that there is sufficient time left for the job to execute fully within the window.

Results

Binary exponential backoff and generalizations. We begin by presenting results on binary exponential backoff. Since a single large job can slow down many small jobs, the performance for heterogeneous job sizes is significantly worse than for unit-sized jobs. We show that it achieves a makespan of $V2^{\Theta(\sqrt{\log n})}$ with error probability polynomially small in n. We next give a variant of exponential backoff that backs off more slowly and yields a makespan of $\Theta(V \log V)$ also with error probability small in n. A key tool is a tight analysis of "fixed-window backoff," where all windows have size $\Theta(V)$. These protocols achieve the specified makespan with error probability polynomially small in the number of jobs.

Size-hashed backoff. The principle result in this paper is a backoff protocol that achieves makespan $O(V \log^3 \log V)$. The main technique is to group jobs by size.

Thus, we "hash" jobs based on their size to specific windows in which they can make run attempts. An explicit construction, based on the modulo hash function results in a makespan of $O(V\sqrt{\log V}\log^2\log V)$. By grouping the job sizes using specially designed "good" hash functions, we obtain makespan $O(V\log^3\log V)$. We use the probabilistic method to show that such hash functions exist. These protocols achieve the specified makespans with error probability polynomially small in $\log V$.

Related Work

The most closely related work is that of Gereb-Graus and Tsantilas [9] (see also [10]) and Bender et al. [11]. Gereb-Graus and Tsantilas [9] show that for unit-size jobs in the batch setting, there is a backoff-backon protocol (which is sometimes called "sawtooth") that achieves an optimal makespan of $O(n)$; a similar backoff-backon approach also appears in Greenberg and Leiserson [10] in the context of routing. Bender et al. [11] analyze fixed backoff, exponential backoff, polynomial backoff, and optimal monotone backoff in the batch setting; they analyze exponential backoff in an adversarial queuing-theory setting. For binary exponential backoff ($W_i = 2^i$) with unit-size jobs, they prove a makespan of $\Theta(n\log n)$. (With variable-length jobs the situation is quite different; see Theorems 3 and 4.) Batch arrivals have been considered by several other authors [12–15] with the goal of routing h-relations, involving multiple channels.

In the wireless-networking literature, this batch problem is known as the ***shopping-cart problem*** [7, 8] and models a shopping cart full of items with RFID tags passing through a sensor all at the same time. Currently implemented protocols are far from achieving the linear makespan described in [9, 10].

2 Traditional Backoff with Variable-Sized Jobs

In this section we analyze classic backoff protocols. We first consider ***fixed backoff***, where the window size is fixed at $\Theta(V)$. (This models the case where an estimate of the volume is known in advance.) We then turn to ***binary exponential backoff***, where the window size repeatedly doubles, i.e., $W_{i+1} = 2W_i$. If a job fits in window, it makes a random run attempt. In both cases the makespan of these strategies is worse for variable-size jobs than for unit-size jobs, with binary exponential backoff significantly worse. We end by giving a faster monotone backoff strategy, whose performance matches fixed backoff to within constant factors, even when the volume V is not known in advance.

Fixed-Volume Backoff

We first analyze the protocol FIXED-BACKOFF$_W$, where the volume V of the jobs is known in advance. FIXED-BACKOFF$_W$ is the windowed protocol in which $W_i = W = \Theta(V)$, where W is the (unchanging) window size throughout the protocol. We first show that FIXED-BACKOFF$_{\Theta(V)}$ has the following lower bound:

Theorem 1. *Let $W = (1+\varepsilon)V$ for any constant $\varepsilon \in \ ^+$. There exists n sufficiently large such that the makespan of* FIXED-BACKOFF$_W$ *is $\Omega(W \log n)$ with error probability polynomially small in n.*

Proof (sketch). Consider an execution with one large job of size $n+1$ and $n-1$ small jobs of size 1. As long as a polylogarithmic number of small jobs remain, at least one small job collides with the large job (w.h.p.); hence the large job does not complete. As long as the large job remains, only a constant fraction of small jobs completes (w.h.p.). Applying a Chernoff bound concludes the proof. □

We now give a matching upper bound for FIXED-BACKOFF$_W$, when $W \geq 3V$.

Theorem 2. *Let $W = \alpha V$, for any $\alpha \geq 3$. Then the makespan of* FIXED-BACKOFF$_W$ *is $O(W \log n)$ with error probability polynomially small in n.*

Proof (sketch). In each round, we argue that a constant fraction of the jobs completes. First, notice that a constant fraction of the jobs are "small," i.e., less than twice the average size. Next, notice that a constant fraction of the small jobs completes: the big jobs can only block $2V$ of the window; the remaining V space is sufficient for each small job to complete with constant probability. □

Notice the difference between fixed backoff in the variable-size and the unit-size case. If all jobs are unit size, then the makespan is $n \lg n \pm O(n)$ with high probability [11]. Moreover, the makespan *improves* when the window size dips slightly below the volume $V = n$, say to $W = 3n/\lg\lg\lg n$, at which point the makespan attains its optimal value of $\Theta(n \log\log n / \log\log\log n)$ [11]. With variable-length jobs, the makespan grows arbitrarily large if $W = V$.

Exponential Backoff

We next analyze binary exponential backoff with variable-size jobs. In binary exponential backoff, $W_i = 2^i$, for $i = 1, 2, \ldots$, and for any job j in the system, j must make a run attempt in window W_i, as long as $t_j \leq W_i$.

Theorem 3. *Consider n jobs with total volume V running binary exponential backoff. The makespan is $V 2^{O(\sqrt{\log n})}$ with error probability polynomially small in n (for sufficiently large n).*

Proof (sketch). In each round we divide the jobs into classes of "small" and "large" jobs. We show that as the window size increases, an increasingly large fraction of jobs completes (with high probability). Specifically, "small" is defined to include an increasingly large fraction of all jobs, and an increasingly large fraction of small jobs complete in each round (with high probability). Within $O(\sqrt{\log n})$ rounds after the window size is W, there are only $O(\log n)$ jobs remaining. Another argument shows that these $O(\log n)$ stragglers complete in the next $O(\sqrt{\log n})$ rounds, with high probability. □

We now give a lower bound on the performance of binary exponential backoff.

Theorem 4. *There exists an instance of* $(c+3)m\ln m+1$ *jobs for which the makespan of exponential backoff is* $\Omega(V2^{\sqrt{\lg V}/2})$ *rounds with probability* $(1-1/m^c)$*, for any* $c>1$*.*

Proof (sketch). Consider an instance with one large job of size m and $(c+3)m\ln m$ small jobs of size 1, resulting in a total volume of $V=(c+3)m\ln m+m$. There are two regimes, which we analyze separately. While $W_i<m$, the *small-window regime*, only small jobs attempt to execute. When $W_i\geq m$, the *large-window regime*, the large job also attempts to execute. No job completes in the small-window regime (w.h.p.) since the small jobs collide with each other. For the first $\Omega(\sqrt{\log m})$ windows of the large-window regime, there exists some small job that collides with the large job in each window (w.h.p.), since the large job blocks a geometrically decreasing fraction of the window. □

Optimized Exponential Backoff

We develop a variant of exponential backoff that achieves better performance by backing off more slowly, nearly matching the performance of fixed backoff.

The idea is to double window sizes (as in exponential backoff) but only after repeating a window of size W $\Theta(\log W)$ times, allowing all jobs to complete when W is an accurate guess of V. Thus, we effectively back off by a factor of only $1+O(1/\log V)$ (rather than 2 as with binary exponential backoff). This algorithm matches the asymptotic performance of FIXED-BACKOFF$_{\Theta(V)}$:

Theorem 5. *There exists a parameter choice for exponential backoff achieving makespan* $O(V\log V)$ *with high probability, i.e., error probability polynomially small in* n*.* □

3 Size-Hashed Backoff

This section describes more efficient backoff protocols that improve on the traditional ones analyzed in Section 2. The main difficulty in dealing with different-sized jobs is that larger jobs are not likely to succeed until enough of the smaller jobs complete. This fact is exploited in Theorem 1's proof, where just one large job interferes with all the other jobs. The approach in this section groups jobs by size so that jobs with different sizes cannot interfere with each other for too long. In particular, we divide jobs into $\lceil\lg V\rceil$ ***job classes*** based on size. Jobs of size t_i belong to the $(\lceil\lg t_i\rceil+1)$th job class.

We first review a "backon" protocol for constant-sized jobs, which forms a subcomponent of our new strategy. We then overview the general strategy for size-hashed backoff. Next, we discuss the mapping "hash" functions (for which the protocol is named). We present the detailed protocol and two specific mapping functions resulting in specific instantiations of size-hashed backoff. Applying our first mapping yields a protocol with makespan $O(V\sqrt{\log V}\log^2\log V)$. We then show the existence of a mapping that achieves $O(V\log^3\log V)$ makespan. Both of these versions achieve the specified makespan with probability $1-1/\log^c V$ for any constant $c>1$ with a linear dependence on c in the makespan.

Backon Protocol for Constant-Sized Jobs

A key component of our strategy is the DESCEND "backon" subprotocol. The protocol (i.e., the participating jobs) takes three parameters: (1) *jclass*, the job class, (2) W, the window size, and (3) r, a number of repetitions. It guarantees that if m processes in the same job class, having total volume V', $V' < W$, all start DESCEND at the same time, then within $O(rW)$ time all the jobs finish with probability $1 - 1/2^r$. The main idea is that once the window has size $3V'$, then a constant fraction of the jobs should complete. At this point, the protocol can "back on," using a window that is a constant fraction smaller. The process continues shrinking the window size until it has decreased to $W/\lg W$. After that point, we repeat the $W/\lg W$-sized window approximately $\lg W$ times. In order to achieve the desired probability, this entire process is repeated r times.

Since a close variant of DESCEND has been previously analyzed by Gereb-Graus and Tsantilas [9], we omit the proof here. (It also follows from Lemma 3.)

Overview of Size-hashed Backoff

As in exponential backoff, size-hashed backoff proceeds by repeated doubling on the estimated volume. We refer to each iteration as a ***round***. The algorithm completes in (or before) the first round in which the estimated volume is sufficiently large ($V' > V$). Each round of the protocol proceeds in ***phases***. When the estimated volume is sufficiently large, in each phase the number of ***nonempty*** job classes—those with jobs remaining—is reduced by a constant fraction.

In the first phase, each job class runs separately. That is, we take a time interval of size $\Theta(rV)$, where $r = \Theta(\log\log V)$ is a number of repetitions for the DESCEND protocol, and divide it into $\lg V$ size-$\Theta(rV/\log V)$ "buckets," one for each job class. Specifically, bucket i is designated for the jobs in job class i (i.e., those jobs j with size $2^{i-2} < t_j \leq 2^{i-1}$). During the ith bucket, each job in the ith job class runs the DESCEND subprotocol for time $\Theta(rV/\log V)$. If the volume in the job class is small enough—specifically, $O(V/\log V)$—then that job class ***completes***, i.e., becomes empty. Since the volume is distributed among various job classes, a constant fraction of the job classes have small enough volume to complete. In particular, a simple counting argument shows that at least $1/2$ of the $\lg V$ job classes have volume at most $2V/\lg V$. We conclude that after $O(rV) = O(V\log\log V)$ time, at least half the job classes are empty.

It would be ideal if, during a second phase, we could allocate buckets for only the nonempty job classes. Since at least half the job classes are empty, we can, in principle, allocate half as many buckets of twice the size and run DESCEND for each bucket. Once again, at least half of the job classes have a small enough volume to complete. After $\lg\lg V$ phases following this process, there is a single nonempty job class in a $\Theta(rV)$-size bucket, and hence this last job class completes. Since each of the phases takes time $\Theta(V\log\log V)$, the resulting makespan is $O(V\log^2\log V)$.

The problem with this approach is that jobs have no *a priori* knowledge as to which job classes become empty during a given phase, and they cannot observe this information. Surprisingly, we can still resurrect the spirit of this idea.

To generalize, we create a mapping from $\lg V$ job classes to a set of buckets smaller than $\lg V$. Given any small set of nonempty job classes, the mapping has the property that a constant fraction are assigned to their own bucket,[3] thus allowing them to complete using DESCEND. To ensure this property, we use extra buckets, resulting in a makespan of $O(V \log^3 \log V)$ for our fastest protocol.

Mapping Job Classes to Buckets

We define the mapping problem more formally. We are given η objects $X = \{x_1, x_2, \ldots, x_\eta\}$ and some integer $m < \eta$. Consider a mapping $F_{m,\eta} : X \to \wp(B)$ of objects to subsets of buckets $B = \{B_1, B_2, \ldots\}$. For example, $F_{m,\eta}(x_1) = \{B_1, B_7, B_{10}\}$ indicates that object x_1 maps to buckets B_1, B_7, and B_{10}.

A mapping is an ***α-good mapping***, with $0 < \alpha \leq 1$, if for all size-m subsets $Y = \{y_1, y_2, \ldots, y_m\} \subseteq X$, there exists a size-$\lceil \alpha m \rceil$ subset of Y in which each object is assigned its own bucket. More formally, $\exists Z \subseteq Y$ where $|Z| = \alpha m$ and $\forall z \in Z, \exists b \in F_{m,\eta}(z)$ s.t. $b \notin \bigcup_{y \in Y \setminus z} F_{m,\eta}(y)$.

This "good mapping" property is exactly what we need for size-hashed backoff. In the backoff setting, $\eta = \lceil \lg V \rceil$ is the number of job classes. In any phase, we maintain an estimate of the number m of nonempty job classes; we do not, however, know which classes are nonempty. We want at least a constant fraction of them to end up assigned to their own buckets. We can then ensure that a constant fraction of job classes complete. For example, if a phase has buckets of size $2rV/m$ (i.e., at most $m/2$ job classes are "too big" to complete) and the mapping is a 3/4-good mapping, then at least $m/4$ of the nonempty job classes must be small enough to complete in a bucket *and* be mapped to a unique bucket.

Our size-hashed backoff algorithm considers good mappings of a simplified form, making the functions easier to think about. Rather than having arbitrary functions from objects to bucket sets, we split the buckets into "collections" of consecutive buckets. Each object is mapped to exactly one bucket in each collection. We construct our mapping $\mathcal{F}_{m,\eta}$ as a sequence of functions $\mathcal{F}_{m,\eta} = \{f_{m,\eta,1}, f_{m,\eta,2}, \ldots, f_{m,\eta,s_{m,\eta}}\}$ such that $f_{m,\eta,i} : X \to B$ maps an object to a single bucket in the ith collection. We define $s_{m,\eta} = |\mathcal{F}_{m,\eta}|$ to be the ***size*** of the set of functions in our mapping $\mathcal{F}_{m,\eta}$. Adding more functions to $\mathcal{F}_{m,\eta}$ increases the chance of achieving α-goodness. We define $r_{m,\eta,i}$ to be the ***range*** of, or the number of buckets used by, the function $f_{m,\eta,i}$.

Size-hashed Protocol

We now give the protocol for size-hashed backoff in more detail, assuming an α-good mapping $\mathcal{F}_{m,\eta} = \{f_{m,\eta,i}\}$. (Pseudocode for the size-hashed protocol is given below.) We argue that all jobs eventually make successful run attempts with probability at least $1 - 1/\lg^c V$ for any constant $c > 1$. The makespan, however,

[3] This property is similar to the collision property of a hash function. It also appears to have close connections to expanders, specifically lossless, bipartite expanders.

```
SIZE-HASHED-BACKOFF(t_i) ▷ t_i is the job size of process i.
 1  V' ← 1
 2  jclass ← ⌈lg t_i⌉ + 1
 3  repeat  ▷ Each iteration is a round.
 4          V' ← 2V'
 5          η ← lg V'
 6          m ← η ▷ m bounds the number of nonempty job classes.
 7          repeat ▷ Each iteration is a phase.
 8                  wsize ← c_1 V'/m ▷ Window size for DESCEND.
 9                  bsize ← c_3 wsize lg lg V' ▷ Bucket for DESCEND iteration.
                    ▷ s_{m,η} is the number of functions in the mapping.
                    ▷ Iterate over subphases/functions.
10                  for i ← 1 to s_{m,η}
11                      do ▷ r_{m,η,i} is the number of buckets used by f_{m,η,i}.
                           ▷ Iterate over buckets.
12                         for bucket ← 1 to r_{m,η,i}
13                             do if f_{m,η,i}(jclass) = bucket
14                                then DESCEND(jclass, wsize, c_3 lg lg V')
15                                else Wait bsize time.
16                  m ← ⌊m/c_2⌋
17            until m = 0 ▷ End loop over phases.
18    until job i executes ▷ Ends the loop over rounds.
```

depends on the size and range of the mapping, so we defer that discussion to the particular variants later in the section.

Recall that size-hashed backoff executes in rounds (lines 3–18), and we repeatedly double the estimated volume in each round (line 4). Each round is divided into phases (lines 7–17), and in each phase we expect a constant fraction of the job classes to complete using the α-good mapping $\mathcal{F}_{m,\eta}$. Each phase is subdivided into ***subphases*** (lines 10–15) which correspond to each function $f_{m,\eta,i}$ in the mapping $\mathcal{F}_{m,\eta}$, so each job class maps to exactly one bucket in each subphase. The α-goodness property guarantees that at least αm of the m nonempty job classes are assigned to unique buckets. The buckets use the geometrically-decreasing DESCEND protocol to ensure that jobs complete when (1) the buckets are large enough, and (2) $\mathcal{F}_{m,\eta}$ assigns a unique bucket (line 14).

Consider the ith phase, during which there should be (at most) m nonempty job classes remaining. During this phase, the protocol creates $s_{m,\eta}$ subphases, where subphase j uses $r_{m,\eta,j}$ buckets of size $bsize = \Theta(rV/m)$ (lines 8–9). Thus, the total length of the ith phase is $\sum_{j=1}^{s_{m,\eta}} r_{m,\eta,j}\Theta(rV/m)$. To understand what these numbers mean, consider the "ideal" mapping in which each job knows exactly which job classes are empty; in this case $s_{m,\eta} = 1$ and $r_{m,\eta,1} = m$, giving a total phase length of $\Theta(rV) = \Theta(V \log\log V)$.

The following theorem states that SIZE-HASHED-BACKOFF completes all the jobs in $\lg V + O(1)$ rounds (i.e., when the window size is $\Theta(V)$). We later analyze the length of each round—and hence the makespan—in the context of the specific family of mappings $\mathcal{F}$, which determines the number of buckets.

Theorem 6. *Suppose n jobs with volume V execute* Size-Hashed-Backoff*, beginning at the same time. Suppose also that $\mathcal{F}$ is an α-good mapping for some constant α. If we set $c_1 = 2/\alpha$, $c_2 = 2/(2-\alpha)$, and $c_3 = c+2$, where c_1, c_2, c_3 are the constants from the pseudocode, then all n jobs complete before the $(\lg V + O(1))$th round with probability at least $1 - 1/\lg^c V$, for any $c \geq 1$.*

Proof (sketch). We show the following invariant holds with sufficient probability: if $V' > V$, then m is an upper bound on the number of nonempty job classes. Initially, there are $\leq m = \eta = \lg V'$ job classes. We proceed by induction. Since the total volume of jobs is $O(V')$, there can be at most $\lfloor m/c_1 \rfloor$ job classes with volume $> \Theta(c_1 V'/m)$. Since $\mathcal{F}$ is α-good, at most $m - \lceil \alpha m \rceil$ of the nonempty job classes *do not* map to their own bucket. Thus, there are at most $m - \lceil \alpha m \rceil + \lfloor m/c_1 \rfloor \leq \lfloor m/c_2 \rfloor$ job classes that are too large or collide. These job classes do not (necessarily) complete during the phase. Any other job class completes during the Descend protocol: each job class completes with probability $1 - 1/\lg^{c_3} V' > 1 - 1/\lg^{c_3} V$. Taking a union bound across all $\lceil \lg V \rceil$ job classes and $\Theta(\log \log V)$ phases maintains the invariant with probability at least $1 - 1/\lg^{c_3 - 2} V$. □

We now provide two α-good mappings, and analyze the resulting performance.

Analysis of a 1-Good Mapping

In this section we present a 1-good mapping based on a simple modulo function, which results in a makespan of $\Theta(V \sqrt{\log V} \log^2 \log V)$.

Let $g_{m,\eta,i}$ be the identity function: $g_{m,\eta,i}(x_j) = j$. (That is, the j^{th} object maps to bucket j.) Recall that each function $g_{m,\eta,i}$ maps objects to exactly one bucket in collection i. Notice that each collection contains η buckets. Let $f_{m,\eta,i}(x_j) = j \pmod i$. Notice that the ith collection contains i buckets. We define a 1-good mapping, parameterized by a variable t (defined later), as follows:

$$\mathcal{F}_{m,\eta} = \begin{cases} \{g_{m,\eta,1}\} & : \text{ if } m > \eta/t \\ \{f_{m,\eta,1}, f_{m,\eta,2}, \ldots, f_{m,\eta,\Theta(m \log \eta)}\} & : \text{ if } m \leq \eta/t \ . \end{cases}$$

Lemma 1. *The functions $\mathcal{F}_{m,\eta}$ are a 1-good mapping.*

Proof (sketch). Notice that if $m > \eta/t$, $\mathcal{F}$ is the identity mapping, which is 1-good. Assume $m \leq \eta/t$. Consider any two objects $x_j \neq x_k \in X$. Consider C prime numbers $p_1, p_2, \ldots, p_C$, each of which is $\geq m \lg \eta$, and suppose by contradiction they collide everywhere, i.e., $f_{m,\eta,p_\ell}(x_j) = f_{m,\eta,p_\ell}(x_k)$ for all $\ell \in 1, 2, \ldots, C$. Then the difference between j and k must be divisble by each of these prime numbers, and hence at least $(m \lg \eta)^C$. Choose $C > \lg \eta / \lg(m \lg \eta)$, implying $(m \lg \eta)^C > \eta$. This is a contradiction, since $|j - k|$ can be at most η. Thus, x_j and x_k can collide in at most $C - 1$ of the functions $\{f_{m,\eta,p_\ell}\}$.

Recall that there are $\Theta(m \log \eta)$ functions $f_{m,\eta,i}$. Thus for a sufficiently large constant in the Θ notation, there are at least $mC = m \lg \eta / \lg(m \lg \eta)$ functions $f_{m,\eta,i}$ in which i is prime and $i \geq m \lg \eta$. For a given set Y of size $\leq m$ and a given object $x_j \in Y$, there must be one of the mC functions in which x_j does not collide with any of the $\leq m$ objects in Y, implying that $\mathcal{F}$ is 1-good. □

We now calculate the running times of each round:

Lemma 2. *The running time for a single round of size-hashed backoff, with 1-good mapping $\mathcal{F}$ is $O(V'\sqrt{\log V'}\log^2\log V')$.*

Proof (sketch). First, consider a phase in which the number of job classes $m > \eta/t$. Recall that in this case, the number of collections $s_{m,\eta} = 1$ and the number of buckets in a collection $r_{m,\eta,1} = \eta$. Thus, the running time of the phase is $r_{m,\eta,1}\, bsize = \Theta(\eta V' \log\log V'/m)$ (lines 8–9). Since m decreases geometrically (by c_2 in each phase), the sum of running times of all phases with $m > \eta/t$ can be bounded by the phase with minimum m, which is thus $O(tV'\log\log V')$.

Consider a phase where $m \le \eta/t$. Then the number of collections $s_{m,\eta} = \Theta(m\log\eta)$ and the number of buckets per collection $r_{m,\eta,i} = i$. Thus, a phase completes in $bsize\sum_{i=1}^{s_{m,\eta}} r_{m,\eta} = bsize\, O(m^2\log^2\eta)$ time. Substituting for *bsize* and η, we have $O(V'\log\log V' m^2\log^2\eta/m) = O(mV'\log^3\log V')$. Since m decreases geometrically, the sum of running times of all phases with $m \le \eta/t$ can be bounded by the phase with maximum m, which is thus $O(V'\log V'\log^3 V'/t)$.

Thus the total duration of a round is $\Theta(tV'\log\log V' + V'\log V'\log^3 V'/t)$. Setting $t = \sqrt{\log V'}\log\log V'$ yields a time of $\Theta(V'\sqrt{\log V'}\log^2\log V')$. □

Theorem 6 shows that size-hashed backoff, when using this 1-good function, terminates of $\lg V + O(1)$ rounds. Together with Lemma 2, we can conclude:

Corollary 1. *Assume that n jobs with volume V begin executing size-hashed backoff with 1-good mapping $\mathcal{F}$ at the same time. Then all n jobs make a successful run attempt in time $O(V\sqrt{\log V}\log^2\log V)$ with probability at least $1 - 1/\lg^c V$, for any $c \ge 1$ and sufficiently large V.* □

Analysis of a 1/2-Good Mapping

Our final version of size-hashed backoff achieves a makespan of $O(V\log^3\log V)$. This algorithm relies on a more efficient α-good mapping, which we show exists using the probabilistic method. The goal of this section is to prove the existence of a 1/2-good mapping where $s_{m,\eta} = \Theta(\log\log V)$ and $r_{m,\eta,i} = \Theta(m)$. This results, as corollary of Theorem 6, in a makespan of $O(V\log^3\log V)$.

Notice that there are three $\log\log V$ factors in the makespan. Two of these come from the general structure of size-hashed backoff: there are $\Theta(\log\log V)$ phases reducing the number of nonempty job classes, and DESCEND runs for $\Theta(\log\log V)$ windows. The third $\log\log V$ factor arises from the number of functions ($s_{m,\eta}$). We first present a preliminary "balls and bins" lemma:

Lemma 3. *Assume you have m balls thrown uniformly at random into cm bins, $c > 15$. Then for some $0 < \delta < 1$, the probability that fewer than $m/2$ bins have exactly one ball is $\le \delta^{cm}$.* □

We can now show that there exist appropriate 1/2-good functions:

Theorem 7. *There exists a set of 1/2-good functions $\mathcal{F}_{m,\eta} = \{f_{m,\eta,i}\}$ where the range $r_{m,\eta,i} = \Theta(m)$ and the number of subphases $s_{m,\eta} = \Theta(\lg\eta)$.*

Proof (sketch). We show the existence of the functions $\{f_{m,\eta,i}\}$ using the probabilistic method. For each η and $m \leq \eta$, for each $i \in [1, s_{m,\eta}]$, choose $f_{m,\eta,i}$ at random: choose $f_{m,\eta,i}(j)$, $j \leq \eta$, uniformly at random from the range $[1, r_{m,\eta,i}]$. We show that with probability > 0, the resulting family of functions is 1/2-good.

First, we calculate for a fixed set Y of size at most m and a fixed $i \in [1 \ldots s_{m,\eta}]$, the probability that $f_{m,\eta,i}$ is 1/2-good. We consider each of the m values $f_{m,\eta,i}(j)$, $j \in Y$, as a ball that is thrown uniformly at random into $r_{m,\eta,i} = \Theta(m)$ bins. By Lemma 3, we know that for some $\delta < 1$, the probability that fewer than 1/2 the "balls" are in their own bin is $\leq \delta^{\Theta(n)}$. That is, the probability that $|\{j \in Y \mid \exists k \in Y, f_{m,\eta,i}(j) = f_{m,\eta,i}(k)\}| > |Y|/2$ is $\leq \delta^{\Theta(m)}$.

Therefore the probability that for all $i \in [1, s_{m,\eta}]$ 1/2-goodness is violated is $\leq \delta^{\Theta(m)s_{m,\eta}} \leq \delta^{\Theta(m \lg \eta)}$, since each function i is selected independently.

Finally, we compute the number of possible sets Y of size m. In particular, there are $\binom{m+\eta}{m} \leq \binom{2\eta}{m}$ ways to distribute m possible jobs over η job classes. This reduces to: $\binom{2\eta}{m} \leq \left(\frac{2e\eta}{m}\right)^m \leq e^m 2^{m \lg \eta}$. We apply a union bound over the possible sets Y: $\Pr[\mathcal{F}_{m,\eta}$ is not 1/2-good$] \leq \delta^{\Theta(m \lg \eta)} e^m 2^{m \lg \eta} < 1$. We conclude that with probability > 0, the randomly chosen $\mathcal{F}_{m,\eta}$ is 1/2-good. □

We now conclude by calculating the makespan as a corollary of Theorem 6:

Corollary 2. *Assume that n jobs with volume V begin executing size-hashed backoff at the same time. Suppose also that we use a 1/2-good mapping $\mathcal{F}$ with sizes $s_{m,\eta} = \Theta(\log \eta)$ and ranges $r_{m,\eta,i} = \Theta(m)$. Then all n jobs make successful run attempts in time $O(V \log^3 \log V)$ with probability at least $1 - 1/\lg^c V$ for any constant $c \geq 1$ and sufficiently large V.*

Proof (sketch). By Theorem 6, all jobs complete by round $\lg V + O(1)$ with appropriate probability. Each phase in the round takes time *bsize* $\sum_{i=1}^{s_{m,\eta}} r_{m,\eta,i} =$ *bsize* $\Theta(m \log \eta)$. Substituting for *bsize* and η yields phase length $\Theta(V \log^2 \log V)$. Summing over $\Theta(\log \log V)$ phases completes the proof. □

4 Conclusion

In this paper, we study randomized backoff protocols when jobs differ in size. We analyze binary exponential backoff and show that it performs poorly, yielding makespan $V 2^{\Theta(\sqrt{\log n})}$. A slower rate of backoff achieves makespan $\Theta(V \log V)$. Our main results are size-hashed backoff protocols, where the backoff strategy depends on the job lengths; we reduce the makespan to only $O(V \log^3 \log V)$.

These results raise many questions. First, what are the lower bounds? Is a linear makespan possible? Next, on a simple channel jobs learn about contention only by making run attempts; what if jobs can listen on the channel without running? Also, a job i learns that a run attempt has failed only after the full t_i time steps. What if jobs learn of failure as soon as a collision occurs, enabling them to abort early? Can exponential backoff and its variants perform better?

This paper considers the batch problem, where jobs arrive at time 0. Ultimately we hope to understand the online problem, where jobs arrive over time.

What can be proved for queuing-theory arrivals? Are there reasonable worst-case models, similar to those in [11], that apply to backoff with different-size jobs?

Acknowledgments. We would like to thank Martin Farach-Colton, Bradley C. Kuszmaul, and Jelani Nelson for helpful conversations and feedback.

References

1. Metcalfe, R.M., Boggs, D.R.: Ethernet: Distributed packet switching for local computer networks. CACM **19**(7) (1976) 395–404
2. Abramson, N.: The ALOHA system — another alternative for computer communications. In: Proc. of AFIPS FJCC. Volume 37. (1970) 281–285
3. Herlihy, M., Moss, J.E.B.: Transactional memory: Architectural support for lock-free data structures. In: Proc. of the 20th Intnl. Conference on Computer Architecture., San Diego, California (1993) 289–300
4. Rajwar, R., Goodman, J.R.: Speculative lock elision: Enabling highly concurrent multithreaded execution. In: Proc. of the 34th Annual Intnl. Symposium on Microarchitecture, Austin, Texas (2001) 294–305
5. Ananian, C.S., Asanović, K., Kuszmaul, B.C., Leiserson, C.E., Lie, S.: Unbounded transactional memory. In: Proc. of the 11th Intnl. Symposium on High-Performance Computer Architecture, San Franscisco, California (2005) 316–327
6. MacKenzie, P.D., Plaxton, C.G., Rajaraman, R.: On contention resolution protocols and associated probabilistic phenomena. JACM **45**(2) (1998) 324–378
7. Juels, A., Rivest, R.L., Szydlo, M.: The blocker tag: Selective blocking of RFID tags for consumer privacy. In: Conference on Computer and Communications Security. (2003) 103–111
8. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. Second edn. John Wiley & Sons (2003) E-book at `books24x7.com`.
9. Geréb-Graus, M., Tsantilas, T.: Efficient optical communication in parallel computers. In: Proc. of the 4th Annual Symposium on Parallel Algorithms and Architectures. (1992) 41–48
10. Greenberg, R.I., Leiserson, C.E.: Randomized routing on fat-trees. Advances in Computing Research **5** (1989) 345–374
11. Bender, M.A., Farach-Colton, M., He, S., Kuszmaul, B.C., Leiserson, C.E.: Adversarial contention resolution for simple channels. In: 17th Annual Symposium on Parallelism in Algorithms and Architectures. (2005) 325–332
12. Greenberg, A.G., Winograd, S.: A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. JACM **32**(3) (1985) 589–596
13. Greenberg, A.G., Flajolet, P., Ladner, R.E.: Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. JACM **34**(2) (1987) 289–325
14. Goldberg, L.A., Jerrum, M., Leighton, T., Rao, S.: Doubly logarithmic communication algorithms for optical-communication parallel computers. SIAM Journal on Computing **26**(4) (1997) 1100–1119
15. Goldberg, L.A., Matias, Y., Rao, S.: An optical simulation of shared memory. SIAM Journal on Computing **28**(5) (1999) 1829–1847