Adversarial Analyses of Window Backoff Strategies for Simple Multiple-Access Channels

Michael A. Bender[1] , Martin Farach-Colton[2], Simai He[1],
Bradley C. Kuszmaul[3], and Charles E. Leiserson[3]

(1) SUNY Stony Brook, (2) Rutgers University, (3) Massachusetts Institute of Technology

***Abstract*— Backoff strategies have typically been analyzed by making statistical assumptions on the distribution of problem inputs. Although these analyses have provided valuable insights into the efficacy of various backoff strategies, they leave open the question as to which backoff algorithms perform best in the worst case or on inputs, such as bursty inputs, that are not covered by the statistical models. This paper analyzes randomized backoff strategies using worst-case assumptions on the inputs.**

Specifically, we analyze algorithms for *simple multiple-access channels*, where the only feedback from each attempt to send a packet is a single bit indicating whether the transmission succeeded or the packet collided with another packet. We analyze a class of strategies, called *window* strategies, where each packet partitions time into a sequence $\langle W_1, W_2, \ldots \rangle$ of windows. Within each window, the packet makes an access attempt during a single randomly selected slot. If its transmission is unsuccessful, it waits for its slot in the next window before retrying.

We use delay-sequence arguments to show that for the *batch* problem, in which n packets all arrive at time 0, if every window has size $W = \Theta(n)$, then with high probability, all packets successfully transmit with makespan $n \lg\lg n \pm O(n)$. We use this result to analyze window backoff strategies with varying window sizes. Specifically, we show that the familiar *binary exponential* backoff algorithm, where $W_k = \Theta(2^k)$, has makespan $\Theta(n \lg n)$, and that more generally, for any constant $r > 1$, the *r-exponential* backoff algorithm in which $W_k = \Theta(r^k)$ has makespan $\Theta(n \lg^{\lg r} n)$. We also show that for any constant $r > 1$, the *r-polynomial* backoff algorithm, in which $W_k = \Theta(k^r)$, has makespan $\Theta((n/\lg n)^{1+1/r})$.

All of these batch strategies are *monotonic*, in the sense that the window size monotonically increases over time. We exhibit a monotonic backoff algorithm that achieves makespan $\Theta(n \lg\lg n/\lg\lg\lg n)$. We prove that this algorithm, whose backoff is superpolynomial and subexponential, is optimal over all monotonic backoff schemes. In addition, we exhibit a simple backoff/backon algorithm, having window sizes that vary nonmonotonically according to a "sawtooth" pattern, that achieves the optimal makespan of $\Theta(n)$.

This research was supported in part by the Singapore-MIT Alliance, Sandia National Laboratories, and NSF Grants ACI-032497, EIA-0112849, CCR-0208670, and CCR-9820879.

We study the *online* setting using an *adversarial queueing model*. We define a (λ, T)-*stream* to be an input stream of packets in which at most $n = \lambda T$ packets arrive during any time interval of size T. In this model, to evaluate a given backoff algorithm (which does not know λ or T), we analyze the worst-case behavior of the algorithm over the class of (λ, T)-streams.

Our results for the online setting focus on exponential backoff. We show that for any arrival rate λ, there exists a sufficiently large interval size T such that the throughput goes to 0 for some (λ, T)-stream. Moreover, there exists a sufficiently large constant c such that for any interval size T, if $\lambda \geq c \lg\lg n/\lg n$, the system is *unstable* in the sense that the arrival rate exceeds the throughput in the worst case. If, on the other hand, we have $\lambda \leq c/\lg n$ for a sufficiently small constant c, then the system is stable. Surprisingly, the algorithms that guarantee smaller makespans in the batch setting require lower arrival rates to achieve stability than does exponential backoff, but when they are stable, they have better response times.

I. Introduction

Backoff is the method of choice for resolving contention in the use of multiple-access channels. The idea of backoff is that whenever a packet experiences a ***collision*** in the use of the channel, it retries, but with a diminished probability of transmission in subsequent time slots. If all packets coöperate in using this strategy and the channel is not oversubscribed, all packets eventually can be transmitted without interference from other packets.

Randomized backoff is perhaps best known in the context of the Ethernet [32] local-area network. When several packets attempt to use the multiple-access Ethernet channel at the same time, a collision occurs, and no packets are successfully transmitted. Ethernet's exponential backoff hardware resolves this contention by retrying packet transmissions with exponentially decreasing frequency. Specifically, whenever an attempted transmission fails due to network contention, the hardware responsible for transmitting the packet doubles the value of a counter, and then it waits for a random amount of time whose expectation is proportional to the value of

the counter before trying to transmit the packet again.

Backoff has proved itself to be an effective and practical method for contention resolution in a myriad of settings besides Ethernet, including radio and satellite networks [1], email retransmission [5], [7], TCP congestion control [36], Sun RPC congestion control [33], HTTP congestion control [8], DHCP retry [42], setting power levels on radio transmitters [44], barrier synchronization in shared-memory multiprocessors [2], optical switching [9], [11], [12], [15], contention resolution in PRAMs [26], [30], randomized routing on fat trees [19], transaction conflict resolution in databases [41] and distributed databases [35], transactional memory access [22], [24], lock conflicts [21], [23], etc.

Given the prominent role played by randomized backoff in computer systems, it is surprising that many aspected of backoff are not yet understood. How do backoff algorithms perform in the worst-case when the arrivals of packets are governed by an adversary rather than a statistical queueing model? How do backoff algorithms behave under the assumption that all packets arrive at the same time versus when they arrive individually over time? What is the proper rate for backing off: exponential, quadratic, or something else? Is there any advantage to "backing on," or should the probability of retransmission simply diminish over time?

The answers to these questions may depend greatly on the model for the multiple-access channel. For example, in some models, packets may have different sizes, taking different amounts of time to transmit. Furthermore, a collision of several packets may allow one or more packet transmissions to succeed, as opposed to all packets failing. It may matter greatly how much information is fed back to the sender when a transmission fails due to a conflict. The backoff algorithm may or may not be able to synchronize transmission attempts based on knowledge of global time.

In this paper, we study the worst-case performance of randomized backoff algorithms for ***simple multiple-access channels***. In this model, all packets take unit time for transmission, and if several packets collide on the channel, none is successfully delivered. The only feedback to the backoff algorithm is the fact that the message was not delivered. Moreover, the algorithm cannot "listen" to the channel and glean information without actually attempting a transmission. Finally, the algorithm cannot exploit knowledge of a global clock in order to synchronize the transmission attempts of different packets. Simple multiple-access channels are useful not only for understanding the shared properties of many conflict resolution systems, but also for exactly modeling some of these situations. For example, satellites such as Aloha are not be able to listen to their channel because of excessive delays [1], while 802.11 [25] wireless links cannot do so because a sender's own transmission is so loud it drowns out the channel information. These systems rely on various kinds of acknowledgments to verify transmission.

We consider backoff strategies in which time is partitioned into a sequence $\langle W_1, W_2, \ldots \rangle$ of ***windows***, and exactly one transmission attempt is made within each window. A backoff strategy is ***monotone*** if $W_i \leq W_{i+1}$.

Packets are injected over time via one of several models. The most common model in the literature is that the packets arrive according to a Poisson distribution with ***arrival rate*** λ. We consider the ***batch model***, in which n jobs all arrive at time 0, and the ***adversarial queueing model*** [6]. We define a $(\boldsymbol{\lambda}, \boldsymbol{T})$***-stream*** to be an input stream of packets in which at most $n = \lambda T$ packets arrive during any time interval of size T. Once again, λ is referred to as the arrival rate, and T is called the ***interval size***.

We use several measures of performance of a backoff strategy with respect to a packet model. For the batch case, we define the ***makespan*** to be the time at which the last packet completes. For a (λ, T)-stream, we define the ***throughput*** to be the number of packets that complete in a window of size T divided by T. The ***response time*** of a packet is the amount of time it is in the system before it successfully transmits.

A channel is said to ***seize*** when its throughput goes to 0. We will say that a backoff strategy is ***unstable*** if there exists a (λ, T)-stream such that the throughput is less than the arrival rate. Otherwise, a backoff strategy is ***stable***.

Most of the prior analytical results for contention resolution assume a statistical queueing-theory model, that is, Poisson arrival of packets. The literature in this area is very rich (see [10] for a nice survey). We mention that Goldberg et al [14] showed that for Poisson arrivals, there is an backoff strategy that achieves $O(1)$ expected response time. Kumar and Merakos had simulation results [27] that bulk arrivals seem to lead to greater stability than Poisson, when using exponential backoff. We will show that the combination of bulk and Poisson-like arrival substantially impaires the stability of exponential backoff.

Several papers have bounds on polynomial backoff rather than exponential [13], [16], [20], [37]. For example, Hastad, Leighton, and Rogoff [20] show that under certain queuing models, quadratic backoff (or any polynomial backoff) is stable for any arrival rate $\lambda < 1$, whereas there is a constant $0 < \lambda_0 < 1$, such that exponential backoff is unstable for any arrival rate $\lambda > \lambda_0$. We will consider the makespan of polynomial backoff in the batch case.

Batch arrivals have been considered by several authors [9], [11], [12], [15], [17], [18], [43], though they

were interested in routing h-relations, involving multiple channels, while we consider a detailed analysis of one channel.

Map and Results

In Section II, we use delay-sequence arguments [39] to show that for the batch arrivals, if every window has size $W = \Theta(n)$, then with high probability[1], all packets successfully transmit with makespan $n \lg\lg n \pm O(n)$.

We use this result to analyze window backoff strategies with varying window sizes. Specifically, in Section III, we show that the binary exponential backoff has makespan $\Theta(n \lg n)$, and that more generally, for any constant $r > 1$, the ***r-exponential*** backoff algorithm in which $W_k = \Theta(r^k)$ has makespan $\Theta(n \lg^{\lg r} n)$. We also show that for any constant $r > 1$, the ***r-polynomial*** backoff algorithm, in which $W_k = \Theta(k^r)$, has makespan $\Theta((n/\lg n)^{1+1/r})$.

All of these batch strategies are monotonic. In Section IV, we exhibit a monotonic backoff algorithm that achieves makespan $\Theta(n \lg\lg n/\lg\lg\lg n)$. We prove that this algorithm, whose backoff is superpolynomial and subexponentional, is optimal over all monotonic backoff schemes. In addition, we exhibit a simple backoff/backon algorithm, having window sizes that vary nonmonotonically according to a "sawtooth" pattern, that achieves the optimal makespan of $\Theta(n)$.

In Section V, we study adversarial packet arrivals. Our results focus on exponential backoff. We show that for any arrival rate λ, there exists a sufficiently large interval size T such that the throughput goes to 0 for some (λ, T)-stream. Moreover, there exists a sufficiently large constant c such that for any interval size T, binary exponential backoff is unstable with respect to some (λ, T)-stream, if $\lambda \geq c \lg\lg n/\lg n$, while there is a sufficiently small c such that binary exponential backoff is stable with respect to any (λ, T)-adversary, if $\lambda \leq c/\lg n$ for a sufficiently small contant c. Surprisingly, the algorithms that guarantee smaller makespans in the batch setting require lower arrival rates to achieve stability than does exponential backoff, but when they are stable, they have better response times.

In Section VI, we wrap up with some analysis of non-window backoff strategies and future work.

II. Fixed backoff

In this section, we analyze a simple backoff algorithm for the batch setting, namely one in which there is no backoff. Specifically, we analyze the fixed backoff algorithm when all windows have the same fixed size which is proportional to the number of packets. We use delay-sequence arguments to prove that with high probability, all packets successfully transmit with makespan $n \lg\lg n \pm O(n)$. This result extends to the situation where all windows have size $\Theta(n)$, but where they need not all have the same size. We shall use these results in Sections III and IV to analyze window backoff strategies with asymptotically varying window sizes.

We shall find it convenient to analyze the fixed backoff algorithm in terms of ***rounds***, where each round consists of a single window. Since we are in the batch setting, the rounds are synchronized across all packets.

We use the technique of "delay sequences" [39], [40] to prove the results of this section. Intuitively, a delay sequence is a "minimal" explanation of why some packet survives for a given number of rounds.

Definition 1: A length-k ***delay sequence*** is an event described by a sequence $\langle S_1, S_2, \ldots, S_k \rangle$ of sets of packets, such that

1. $S_{i+1} \subseteq S_i \quad (1 \leq i < k)$;
2. $2|S_{i+1}| \geq |S_i| \quad (1 \leq i < k)$;
3. $|S_k| = 2$.

We say that a delay sequence $(S_1, S_2, \ldots, S_k)$ ***occurs*** if in round i, each packet in set S_i collides with another packet in S_i, thus ***surviving*** round i. The ***volume*** of the delay sequence is defined to be $S_{\text{sum}} = |S_1| + \cdots + |S_k|$, and the ***base*** of the delay sequence is S_1.

We first prove the upper bound of $n \lg\lg n + O(n)$ for the fixed backoff algorithm. Although it is straightforward to show a makespan of $\Theta(n \lg\lg n)$ with high probability (proving this result is an exercise in [34]), we shall see in Section III that backoff protocols can be exquisitely sensitive to constants. Consequently, our analysis relegates asymptotic notation to second-order terms.

Theorem 2: Consider a batch instance in which all n packets have fixed linear window size $W \geq 3e^3 n$. Then, all packets transmit successfully in time $n \lg\lg n + cn$ with probability at least $1 - n^{-2^c+2}$.

Proof Sketch. The full proof argues that if some packet p survives k rounds, then some length-k delay sequence occurs. It then shows that the probability of a given length-k delay sequence $(S_1, S_2, \ldots, S_k)$ occurring is at most $(e|S_1|/2W)^{S_{\text{sum}}/2}$. The number of distinct length-k delay sequences for given values of $|S_1|$ and S_{sum} can then be bounded as

$$\text{NUMDS}(|S_1|, S_{\text{sum}}) \quad \leq \quad \left(\frac{ne^2(S_{\text{sum}} + |S_1|)}{|S_1|^2}\right)^{|S_1|} .$$

The sets $S_1, \ldots, S_k$ obey the following size restrictions:

1. $|S_1| \leq 2^k$;
2. $S_{\text{sum}} \geq 2\,(|S_1| - 1 + k - \lg|S_1|)$.

[1]Define ***high probability*** to mean with probability at least $1 - O(n^{-O(1)})$. We say that a parameterized event E_p occurs ***with high probability*** if for any constant $c > 0$ there exists a valid choice of parameter p such that $\Pr\{E_p\} \geq 1 - n^{-c}$.

Let PROBDS$(|S_1|, S_{\text{sum}})$ represent the probability that a given delay sequence occurs having the values $|S_1|$ and S_{sum}. The probability that *any* length-k delay sequence occurs is at most

$$\sum_{|S_1|=2}^{2^k} \sum_{S_{\text{sum}}=2(|S_1|-1+k-\lg|S_1|)}^{k|S_1|} \text{NUMDS}(|S_1|, S_{\text{sum}}) \text{PROBDS}(|S_1|, S_{\text{sum}}).$$

Observe that the largest term in the sum occurs when the volume is as small as possible and the base is as large as possible. Plugging in $W \geq 3e^3 n$ and $k = \lg\lg n + c$ completes the proof. □

The bounds from Theorem 2 hold even if the windows have different sizes in different stages, as long as they are sufficiently large.

Corollary 3: Consider a uniform batch instance in which the packet windows have size $W \geq 3e^3 n$, $(i \geq 1)$. All packets transmit successfully in at most $\lg\lg n + c$ windows with probability at least $1 - n^{-2^c+2}$. □

We now use a delay-sequence argument[2] to prove that if all the n packets have a fixed linear window size $W = \Theta(n)$, then some packet requires time $n\lg\lg n - O(n)$ with high probability to transmit successfully.

The delay sequence for the lower bound is as follows:

Definition 4: A length-k ***delay sequence*** is an event described by ordered pair (T_k, J), where

1. Tree T_k is a complete binary tree of height k; height is defined so that a single node has height 1.
2. Ordered set J is a sequence of 2^k distinct packets.
3. Each node $u \in T_k$ is an ordered pair of packets $(u_{\text{left}}, u_{\text{right}})$.
4. Let nodes v and w be the left and right children respectively of node u. Then, $u_{\text{left}} = v_{\text{left}}$ and $u_{\text{right}} = w_{\text{left}}$.

A length-k delay sequence (T_k, J) ***occurs*** if the following conditions hold:

1. For each node $u \in T_k$ of height h, packets u_{left} and u_{right} collide with each other in round h.
2. Moreover, for each node $u \in T_k$ of height h, packets u_{left} and u_{right} collide with *no* other packets (including packets not in J) in this round.
3. For each node $u \in T_k$ of height h, packet u_{right} completes in round $h+1$, i.e., in round $h+1$ packet u_{right} collides with no other packets.

Consider two delay sequences (T_k, J) and (T'_k, J) that are identical except that the order of the packets in the root node of the tree is reversed; (T_k, J) and (T'_k, J) describe the same computational event. We say that two delay sequences are ***distinct*** if they described different events. We say that two delay sequences (T_k, J) and $(T'_{k'}, J')$ ***overlap*** if they share common packets, i.e., $J \cap J' \neq \emptyset$.

[2]Note to reviewers: The authors would be interested to learn of any other lower bounds proved using a delay-sequence argument.

Theorem 5: For n packets with window size $2n \leq W \leq 4n$, there exists one packet not completed after $k = \lfloor \lg\lg n \rfloor - 3$ rounds with probability at least $1 - 4n^{-3/8}$.

Proof Sketch. The full proof argues that if a length-k delay sequence occurs, then some packet p survives (exactly) $k+1$ rounds. It then shows that the probability that a given delay sequence (T_k, J) occurs is at least

$$\left(\frac{1}{W}\right)^{2^k-1} \left(\frac{W-n}{W}\right)^{2^{k+1}-2}.$$

The number of distinct length-k delay sequences is the number of ways of selecting set J with order divided by 2, and this quantity can therefore be bounded as $(1/2)\, n!/(n-2^k)!$.

Because a lower bound is being established, the proof cannot use the union bound as in a majority of delay-sequence arguments, including the proof of Theorem 2. Instead, the full proof identifies discrete events that can be added together *with no overestimation*. Specifically, the probability that exactly one height-k delay sequence occurs is exactly

$$\sum_{(T_k, J)} \Pr\{\text{Only delay sequence } (T_k, J) \text{ occurs}\}.$$

Definition 4 is structured to ensure that if two delay sequences (T_k, J) and $(T'_{k'}, J')$ overlap, then at most one of (T_k, J) and $(T'_{k'}, J')$ can occur. Consequently, when two delay sequences both occur, they have no common packets.

Suppose that a height-k delay sequence (T_k, J) occurs. The probability that a second height-k delay sequence (T'_k, J') occurs is at most the probability of there exists a delay sequence of height k for $n - 2^k$ packets (again with window size W). This probability is less or equal to the probability of having at least one packet not completed after k rounds starting from $n - 2^k$ packets with window size W. This last probability is less or equal than the probability p of having at least one packet not completed after k rounds starting from n packets and with window size W.

For each delay sequence (T_k, J), we have

$$\Pr\{(T_k, J) \text{ occurs and no other delay sequences occur}\} \geq (1-p)\left($$

Let p be the probability that at least one packet is not completed after k rounds. Because there are $n!/2(n-2^k)!$ delay sequences, we have

$$\begin{aligned} p &\geq \Pr\{\text{At least one delay sequence of height } k \text{ occurs}\} \\ &\geq \Pr\{\text{Exactly one delay sequence of height } k \text{ occurs}\} \\ &\geq \frac{n!}{2(n-2^k)!}(1-p)\left(\frac{1}{W}\right)^{2^k-1}\left(\frac{W-n}{W}\right)^{2^{k+1}-2}. \end{aligned}$$

The full proof simplifies further to obtain

$$p \geq (1-p)\frac{n}{4}\left(\frac{n}{8W}\right)^{2^k-1}.$$

Observe that $n/(8W) \geq 2^{-5}$ and $k = \lfloor \lg\lg n \rfloor - 3$. Thus, we have $p \geq (1-p)2^{(3/8)\lg n - 2}$, implying that

$$\begin{aligned} p &\geq 1 - 4n^{-3/8} \\ &= 1 - o(1)\ . \end{aligned}$$

□

III. Exponential and polynomial backoff

This section analyzes exponential and polynomial backoff strategies in the batch setting. We show that the familiar ***binary exponential*** backoff algorithm, in which every packet's kth window has size $W_k = \Theta(2^k)$, has makespan $\Theta(n \lg n)$ with high probability. More generally, we show that for any constant $r > 1$, the ***r-exponential*** backoff algorithm, in which $W_k = \Theta(r^k)$, has makespan $\Theta(n(\lg n)^{\lg r})$ with high probability. We also show that for any constant $r > 1$, the ***r-polynomial*** backoff algorithm, in which $W_k = \Theta(k^r)$, has makespan $\Theta((n/\lg n)^{1+1/r})$ with high probability. Thus, exponential backoff is superior to polynomial backoff in the batch setting.

Theorem 6: Binary exponential backoff has makespan at most $6e^3 2^{c+1} n \lg n$ with probability at least $1 - n^{-2^c+2}$, and makespan at least $n \lg n/196$ with probability at least $1 - 1/(1+\sqrt{n}/2)$.

Proof Sketch. The main part of the analysis begins after the first $n/2$ steps, after which the window size is $\Theta(n)$. At most $n/2$ packets can be transmitted during this interval, although small window sizes mean that many fewer packets are in fact transmitted; for the upper bound we can assume that no packets transmit. We show that once the window size is $\Theta(n)$, only $\lg\lg n + O(1)$ rounds are necessary and sufficient to transmit all packets. The upper bound follows from Corollary 3; the larger window sizes ensure that no more rounds are necessary than in Theorem 2. □

It may seem surprising that the number of rounds is no *fewer* than $\lg\lg n + O(1)$ even though the window sizes are exponentially increasing; the proof of this claim is similar to the proof of Theorem 5. The delay-sequence argument gives intuition why: most of the nodes in the tree-structure of the delay sequence are near the leaves, meaning that the probability that the delay sequence occurs only increases marginally.

Observe the exquisite sensitivity of exponential backoff to the constants: an additive constant in the number of rounds translates to a multiplicative constant in the makespan. Thus, if instead of doubling window sizes in each round, we quadrupled the window sizes, then the makespan would become $\Theta(n \log^2 n)$. More generally, we have the following corollary.

Corollary 7: Any r-exponential backoff algorithm has makespan $\Theta((n/\lg n)^{1+1/r})$ with high probability. □

An alternative backoff strategy is quadratic backoff or, more generally, polynomial backoff. It turns out that quadratic backoff is too slow in general, having makespan $\Theta((n/\lg n)^{3/2})$ with high probability. The following theorem proves this result for the general case of polynomial backoff.

Theorem 8: For any constant $r > 1$, the r-polynomial backoff algorithm, in which $W_k = \Theta(k^r)$, has makespan $\Theta((n/\lg n)^{1+1/r})$ with high probability.

Proof Sketch. Because polynomial backoff increases window sizes slowly, it is not as sensitive to constants as exponential backoff. Specifically, there are $\Theta(W^{1+r})$ time steps before the window sizes reach W. While the window size is $cn/\lg n$, for sufficiently small c, few packets successfully transmit. For larger c, however, the probability of a successful transmission increases and all packets transmit successfully before the window size increases by a constant factor. □

Thus, for quadratic backoff in particular, and for polynomial backoff in general, the dominant cost is waiting until the window size grows sufficiently large.

IV. Optimal backoff for the batch setting

All of the batch strategies we have seen thus far are ***monotonic***, in the sense that the window sizes increase monotonically over time, but none are optimal, even over the set of all monotonic backoff strategies . In this section, we exhibit a monotonic backoff algorithm that achieves makespan $\Theta(n \lg\lg n/\lg\lg\lg n)$. We prove that this "log-log iterated" backoff algorithm, which is superpolynomial and subexponential, is optimal over all monotonic backoff schemes. In addition, we exhibit a nonmonotonic backoff/backon algorithm that achieves the optimal makespan of $\Theta(n)$. This algorithm, which is simple in both design and analysis, has window sizes that vary nonmonotonically according to a "sawtooth" pattern.

The ***log-log iterated backoff algorithm*** behaves like exponential backoff in that it repeatedly doubles its window size, but it stays with each window size W for $2\lg\lg W$ rounds before doubling.

Theorem 9: Log-log iterated backoff has makespan $O(n \lg\lg n/\lg\lg\lg n)$ with high probability.

Proof Sketch. The full proof divides time into rounds, where each round contains exactly one window. When the windows are smaller than $cn/\lg\lg\lg n$, for $c < 2$, we need not assume any successful transmissions: the few packets that successfully transmit only decrease the makespan.

The main part of the analysis begins when the window size is at least $cn/\lg\lg\lg n$, for c bounded above 2 by a constant. We claim that all the packets transmit successfully before the window size doubles. Specifically, after $\lg\lg n$ further rounds there are at most $n/\ln\ln\ln n$ packets left in the system with high probability because the probability of a transmission is one over an exponential in $\Theta(\lg\lg\lg n)$. By Theorem 2, these remaining packets transmit within the next $\lg\lg n$ rounds with high probability. □

We now show that any monotone strategy has makespan $\Omega(n\lg\lg n/\lg\lg\lg n)$ with high probability. This lower bound uses a modification of Theorem 5 and a counting argument.

Theorem 10: Any monotone window backoff strategy for n packets has makespan at least $\Omega(n\lg\lg n/\lg\lg\lg n)$ with high probability.

Proof Sketch. The full proof first establishes that without loss of generality, the expected number of packets that transmits per timestep must be $O(\lg\lg\lg n)$ or the probability of a collision is too great and few packets transmit successfully. Consequently, at least a constant fraction of the packets must have window size $\Omega(n/\lg\lg\lg n)$. For a sufficiently small constant c, we can let any packet with window size smaller than $cn/\lg\lg\lg n$ transmit successfully within even accounting for the increase in makespan from collisions with these packets. A constant fraction of packets still remain in the system. All packets must have maximum window size $O(n\lg\lg n/\lg\lg\lg n)$ in order to have hope of achieving the bounds.

Packets p and p' have ***approximately synchronized*** windows if for all i, the packets' ith windows are at least 95% overlapping. The full proof uses a delay-sequence argument to establish that for any constant c, if $\Omega(n/(\lg\lg n)^{O(\lg\lg^c n)})$ packets have approximately synchronized windows of size at most $O(n\lg\lg n)$, then with high probability $\Omega(\lg\lg n)$ rounds are necessary to transmit all jobs.

The full proof uses an accounting argument to show that while most packet's windows need not be synchronized, there exists a sufficiently large set of packets whose windows are approximately synchronized. First, the proof divides time into $O(\lg\lg n)$ epochs of size $O(n/\lg\lg\lg n)$ where each epoch is a constant factor smaller than the minimum window size. The proof also divides packet window sizes into classes ranging in size from $\Theta(n/\lg\lg\lg n)$ to $\Theta(n\lg\lg n/\lg\lg\lg n)$. Windows in the same class differ in size by most a $(1 + O(1/\lg\lg n))$ factor, implying that there are $O(\lg\lg n\lg\lg\lg n)$ classes. The proof then argues that if for all i, packets p and p' have their ith windows in the same class and their first windows are approximately synchronized, then p and p' have *all* their windows approximately synchronized.

The proof then counts the number of choices each packet has for all of its window sizes: for each of the $O(\lg\lg n)$ epochs, there are $\Theta(\lg\lg n\lg\lg\lg n)$ choices, yielding a total of $O((\lg\lg n)^{O(\lg\lg n\lg\lg\lg n)})$ possibilities. Because there are $\Theta(n)$ packets, some set of at least $\Omega(n/(\lg\lg n)^{O(\lg\lg n\lg\lg\lg n)})$ packets agree on all choices and therefore are approximately synchronized. Because there are $\Omega(\lg\lg n)$ rounds for these packets and each round has size $\Omega(n/\lg\lg\lg n)$, the theorem follows. □

In fact, there exists an optimal nonmonotone backoff algorithm with $\Theta(n)$ makespan, which we call "sawtooth" backoff, because the jagged increases and decreases in transmission probabilities is reminiscent of the structure of a saw blade. This simple backoff algorithm is similar to those proposed in [9], [19].

The sawtooth backoff strategy involves a doubly nested loop. The outer loop performs repeated doubling to "guess" a window size W proportional to the number n of competing messages. The inner loop consists of $\Theta(\lg W)$ phases of "backon," where the window size reduces from the guess W by a constant factor for each phase.

Theorem 11: There is a sawtooth backoff strategy for the batch setting having makespan $\Theta(n)$.

Proof Sketch. We first examine the phases corresponding to the first guess W that equals or exceeds the number n of packets, that is, $n \le W < 2n$. Since, after the first phase, at most a constant fraction of the n messages remain, the second phase shrinks the window by a slightly smaller constant factor, and the algorithm repeats. The intuition is that every phase uses a window that tracks closely the number of packets not yet successfully transmitted. There are $\Theta(\lg n)$ phases, but since the window size shrinks geometrically, the makespan is $\Theta(n)$. The full proof provides the complete combinatorial argument.

Of course, the sawtooth algorithm doesn't know n, but its outer loop repeatedly doubles its previous guess. Once the guess exceeds n, all packets complete as before. The repeated doubling only increases the makespan by a constant factor. □

V. Online backoff

We now turn to the ***online*** setting which we analyze using an ***adversarial queueing model***. Our results focus on exponential backoff and log-log iterated backoff. We show that for any arrival rate λ, there exists a sufficiently large interval size T such that the throughput goes to 0 for some (λ, T)-stream. Moreover, there exists a sufficiently large constant c such that for any interval size T, if $\lambda \geq c \lg\lg n / \lg n$, the system is ***unstable*** in the sense that the arrival rate exceeds the throughput in the worst case. If, on the other hand, we have $\lambda \leq c / \lg n$ for a sufficiently small constant c, then the system is stable. Surprisingly, the algorithms that guarantee smaller makespans in the batch setting require lower arrival rates to achieve stability than does exponential backoff, but when they are stable, they have better response times.

For the online setting, arrivals are determined by a worst-case (λ, T)-stream which injects at most $n = \lambda T$ packets in any window of size T. We say that a backoff strategy is ***unstable*** if there exists a (λ, T)-stream such that the rate at which packets complete, that is, the throughput, is lower than the arrival rate. Otherwise, a backoff strategy is ***stable***.

We give bounds on the stability of exponential backoff as follows.

Theorem 12: The is a sufficiently small constant c so that exponential backoff is stable for any (λ, T)-stream, as long as $\lambda \leq c / \lg n$.

Proof Sketch. We show by induction that every packet transmits successfully in $c' \lg n$ attempts, for $c' < c$, with high probability Thus, the contention remains a constant less than 1, and the throughput matches the arrival rate.

The base case tells most of the story. Consider the first packet p that has survived until its window is of size at least T, which happens after $\lg T$ run attempts. Now, consider the probability of success in each further run attempt. Packet p survives if its transmission attempt collides with a subsequently injected packet. Of these small packets, packets with windows of size at most T each make at most $\lg T = \Theta(\lg n)$ run attempts before having large windows for a total of $\Theta(n \lg n) = \Theta(T)$ run attempt in any window of size T. For large enough c, these packets fill an arbitrarily small constant fraction of time slots with their run attempts. Similarly, we count the number of packets with large window sizes and their number of run attempts. While there can be more large-window packets in the system than small-window packets, large packets make transmission attempts less frequently. We conclude that each of p's run attempts succeeds with constant probability, thus giving the desired bound.

The inductive step handles packets injected before the packet p, of interest. But, by induction, there are few large packets that have survived to interfere with p's run attempts, and a straightforward counting argument with Chernoff bounds finishes approved . □

We now show a lower bound.

Theorem 13: The exists a sufficiently large constant c so that exponential backoff is unstable for any (λ, T)-stream, as long as $\lambda \geq c \log\log n / \log n$.

Proof Sketch. The injection strategy is constructed as follows: Every T steps, $n/2$ packets are injected; every $2 \log n / c \log\log n$ steps, one packet is injected. Thus, n packets are injected in any window of size T. We call the first $n/2$ packets the *bolus* and the following packets the *drip*.

The proof proceeds by showing that in any window of T steps beginning with the injection of the bolus $O(n / \lg\lg n)$ packets complete. To bound the number of completed packets, we show that the contention remains above $\lg\lg n$ with high probability, and thus packets complete on average every $\Omega(\lg^c n)$ steps, where constant c can be is a function of the constant in the arrival rate.

To bound the contention, we show that by the time the contention due to the bolus reaches $\lg n$ – during which time, essentially no packets complete – enough drip packets have been injected to contribute $c' \lg\lg n$ to the contention, for some $c' > 1$. We use Chernoff bounds, with a few technical twists, to show that the contention due to drip packets stays well above $\lg\lg n$ with high probability, even though some packets successfully transmit. □

VI. Conclusion

Our results for window backoff on simple multiple-access channels can be extended in two ways. First, in some settings, one may wish to use a backoff strategy that is not based on windows. Second, one may wish to employ backoff in a multiple-access channel where more information is available from an unsuccessful transmission than is provided by the simple multiple-access channel. To conclude, we discuss the possiblities for future results along both these lines.

In the model of a simple multiple-access channel, a backoff strategy can be viewed as a sequence $p_0, p_1, \ldots$ of random variables, where p_t is the probability that an as-yet undelivered packet is transmitted on the tth step after its arrival. We say that a backoff strategy is ***Bernoulli*** if p_t is a function only of t, in which case all p_t are mutually independent. How do Bernoulli backoff strategies compare with window strategies?

It turns out that in the batch case, any monotone Bernoulli backoff algorithm has makespan $\Omega(n \lg n / \lg\lg n)$ with high probability, even when

n is known. This result is tight, because monontone Bernoulli backoff strategies exist that match this bound, even without knowing n. Thus, the log-log interated backoff algorithm, which is optimal for monotone window backoff, offers smaller makespans by a factor of $\Theta((\lg\lg\lg n)/(\lg\lg n)^2)$ over the optimal monotone Bernoulli backoff. For nonmonotone Bernoulli backoff, however, a Bernoulli sawtooth algorithm can achieve the same $\Theta(n)$ makespan as the window sawtooth algorithm.

With respect to other models of multiple-access channels, the opportunities for future research seem rife. For example, what happens if a sender gets more information back from the channel than just success or failure? For example, a sender may know when others are transmitting, as in the 802.11 wireless standard [25]. Error codes can be used to determine whether a packet collides with exactly one other packet or several. Can this information be used to improve performance? What happens if during a collision, one of the packets succeeds, as in [38]? What happens if packets take different amounts of time to transmit?

These questions appear particularly relevant for online settings, which occur more commonly in practice than batch settings. Currently, most computer engineers rely on simulations, not theoretical analyses, to gain confidence in backoff algorithms. As a consequence, systems employing backoff are generally nonalgorithmic, in the sense that performance is not guaranteed, not even statistically. Consequently, systems can exhibit wildly unpredictable performance, making it difficult or impossible to meet real-time constraints. We are optimistic that further research on backoff algorithms, using techniques such as adversarial queueing theory, will lead to more stable and higher-performing computer systems.

References

[1] N. Abramson. The Aloha system. In N. Abramson and F. Kuo, editors, *Computer-Communication Networks*, pages 501–518. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[2] A. Agarwal and M. Cherian. Adaptive backoff synchronization techniques. In *The 16th Annual International Symposium on Computer Architecture*, pages 396–406, Jerusalem, Israel, May 1989. ACM SIGARCH Computer Architecture News, Volume 17, Number 3, June 1989.

[3] D. J. Aldous. Ultimate instability of exponential back-off protocol for acknowledgment-based transmission control of random access communication channels. *IEEE Transactions on Information Theory*, IT-33(2):219–223, Mar. 1987.

[4] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, Oct. 1991.

[5] D. J. Bernstein. qmail. An email Message Transfer Agent (software), available from `http://cr.yp.to/qmail.html`, JUN 1998.

[6] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM (JACM)*, 48(1):13–38, 2001.

[7] B. Costales and E. Allman. *Sendmail*. O'Reilly, third edition, Dec. 2002.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol—http/1.1. Interent RFC 2616, June 1999.

[9] 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)*, pages 41–48, 1992.

[10] L. A. Goldberg. Notes on contention resolution. `http://www.dcs.warwick.ac.uk/~leslie/contention.html`, viewed Oct. 2003., Oct. 200.

[11] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 300–309, 1993.

[12] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. Doubly logarithmic communication algorithms for optical-communication parallel computers. *SIAM Journal on Computing*, 26(4):1100–1119, Aug. 1997.

[13] L. A. Goldberg and P. D. MacKenzie. Analysis of practical backoff protocols for contention resolution with multiple servers. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 554–563, 1996.

[14] L. A. Goldberg, P. D. MacKenzie, M. Paterson, and A. Srinivasan. Contention resolution with constant expected delay. *Journal of the ACM (JACM)*, 47(6):1048–1096, Nov. 2000.

[15] L. A. Goldberg, Y. Matias, and S. Rao. An optical simulation of shared memory. *SIAM Journal on Computing*, 28(5):1829–1847, Oct. 1999.

[16] J. Goodman, A. G. Greenberg, N. Madras, and P. March. Stability of binary exponential backoff. *Journal of the ACM*, 35(3):579–602, July 1988.

[17] A. G. Greenberg, P. Flajolet, and R. E. Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *Journal of the ACM*, 34(2):289–325, Apr. 1987.

[18] 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. *Journal of the ACM*, 32(3):589–596, July 1985.

[19] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. *Advances in Computing Research*, 5:345–374, 1989.

[20] J. Hastad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. In *Proceedings of the Ninteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 241–253, New York, New York, May 1987. Extended abstract.

[21] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Transactactions on Programming Languages and Systems*, 15(5):745–770, Nov. 1993.

[22] 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. (Also published as ACM SIGARCH Computer Architecture News, Volume 21, Issue 2, May 1993.)*, pages 289–300, San Diego, California, 1993.

[23] M. P. Herlihy, V. Luchangco, and M. Moir. Obstruction-free synchronization: Double-ended queues as an example. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 522–529, Providence, Rhode Island, May 2003.

[24] M. P. Herlihy, V. Luchangco, M. Moir, and W. M. Scherer III. Software transactional memory for dynamic-sized data structures. In *Proceedings of the Twenty-Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 92–101, Boston, Massachusetts, July 2003.

[25] IEEE 802.11 Working Group. *ANSI/IEEE Std. 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society, 1999.

[26] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient

PRAM simulation on a distributed memory machine. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 318–326, Victoria, British Columbia, Canada, May 1992.

[27] P. Kumar and L. Merakos. Distributed control of broadcast channels with acknowledgement feedback: Stability and performance. In *Proceedings IEEE Conference on Decision and Control*, Las Vegas, Nevada, Dec. 1984.

[28] S. S. Lam. Some satellite simulation results. ARPA Network Information Center, Stanford Research Institute, Arpa Satellite System (ASS) Note 48 (NIC 17655), Aug. 1973.

[29] S. S. Lam. *Packet Switching in a Multi-Access Broadcast Channel with Application to Satellite Communication in a Computer Network*. PhD thesis, Computer Science Department, University of California at Los Angeles, Mar. 1974. Published as Technical Report UCLA-ENG-7429, April 1974, and as `http://www.cs.utexas.edu/users/lam/Vita/UCLA/`.

[30] P. D. MacKenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. *Journal of the ACM*, 45(2):324–378, Mar. 1998.

[31] C. U. Martel and T. P. Vayda. The complexity of selection resolution, conflict resolution and maximum finding on multiple access channels. In J. H. Reif, editor, *VLSI Algorithms and Architectures, 3rd Aegean Workshop on Computing (AWOC '88)*, volume 319 of *Lecture Notes in Computer Science*, pages 401–410, Corfu, Greece, June 28–July 1 1988. Springer Lecture Notes in Computer Science.

[32] R. M. Metcalfe and D. R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[33] S. Microsystems. RPC: Remote procedure call protocol specification version 2. Internet RFC 1057, June 1988.

[34] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, England, June 1995.

[35] F. M. Pittelli and H. Garcia-Molina. Reliable scheduling in a TMR database system. *ACM Transactions on Computer Systems*, 7(1):25–60, Feb. 1989.

[36] J. Postel. Transmission control protocol. Internet RFC 793, Sept. 1981.

[37] P. Raghavan and E. Upfal. Stochastic contention resolution with short delays. *SIAM Journal on Computing*, 28(2):709–719, Apr. 1999.

[38] R. Rajwar and J. R. Goodman. Transactional lock-free execution of lock-based programs. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 5–17, San Jose, California, Oct. 5–9 2002.

[39] A. G. Ranade. How to emulate shared memory. *Journal of Computer and Systems Sciences*, 42(3):307–326, 1991.

[40] A. G. Ranade. The delay sequence argument. In *Handbook of Randomized Algorithms*, chapter 1. Kluwer Academic Publishers, 2001.

[41] The Berkeley Database version 2. Software available from `http://sleepycat.com`, 1997.

[42] Y. T'Joens, C. Hublet, and P. De Schrijver. DHCP reconfiguration extension, Dec. 2001.

[43] D. E. Willard. Log-logarithmic protocols for resolving ethernet and semaphore conflicts. In *Proceeedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 512–521, Washington, D.C., Apr. 30–May 2 1984.

[44] Intelligent amplifiers. `http://www.wiseband.com/Intelligant_amplifiers.pdf` (viewed October 2003), Sept. 2002.