Adversarial Contention Resolution for Simple Channels

Michael A. Bender
Department of Computer Science
SUNY Stony Brook
Stony Brook, NY 11794, USA
bender@cs.sunysb.edu

Martin Farach-Colton
Department of Computer Science
Rutgers University
Piscataway, NJ 08855, USA.
farach@cs.rutgers.edu

Simai He
Department of Computer Science
SUNY Stony Brook
Stony Brook, NY 11794, USA
simaihe@cs.sunysb.edu

Bradley C. Kuszmaul
MIT Computer Science and
Artificial Intelligence Laboratory
The Stata Center, 32 Vassar St.
Cambridge, MA 02139, USA
bradley@mit.edu

Charles E. Leiserson
MIT Computer Science and
Artificial Intelligence Laboratory
The Stata Center, 32 Vassar St.
Cambridge, MA 02139, USA
cel@mit.edu

ABSTRACT

This paper analyzes the worst-case performance of randomized backoff on simple multiple-access channels. Most previous analysis of backoff has assumed a statistical arrival model.

For *batched arrivals*, in which all n packets arrive at time 0, we show the following tight high-probability bounds. Randomized binary exponential backoff has makespan $\Theta(n \lg n)$, and more generally, for any constant r, r-exponential backoff has makespan $\Theta(n \log^{\lg r} n)$. Quadratic backoff has makespan $\Theta((n/\lg n)^{3/2})$, and more generally, for $r > 1$, r-polynomial backoff has makespan $\Theta((n/\lg n)^{1+1/r})$. Thus, for batched inputs, both exponential and polynomial backoff are highly sensitive to backoff constants. We exhibit a monotone superpolynomial subexponential backoff algorithm, called *loglog-iterated backoff*, that achieves makespan $\Theta(n \lg\lg n/\lg\lg\lg n)$. We provide a matching lower bound showing that this strategy is optimal among all monotone backoff algorithms. Of independent interest is that this lower bound was proved with a delay sequence argument.

In the adversarial-queuing model, we present the following stability and instability results for exponential backoff and loglog-iterated backoff. Given a (λ, T)-stream, in which at most $n = \lambda T$ packets arrive in any interval of size T, exponential backoff is stable for arrival rates of $\lambda = O(1/\lg n)$ and unstable for arrival rates of $\lambda = \Omega(\lg\lg n/\lg n)$; loglog-iterated backoff is stable for arrival rates of $\lambda = O(1/(\lg\lg n \lg n))$ and unstable for arrival rates of $\lambda = \Omega(1/\lg n)$. Our instability results show that bursty input is close to being worst-case for exponential backoff and variants and that even small bursts can create instabilities in the channel.

This research was supported in part by the Singapore-MIT Alliance and NSF Grants ACI-0324974, CNS-0305606, EIA-0112849, CCR-0208670, and CCR-9820879.

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

Categories and Subject Descriptors: C.4 **[Performance of Systems]**: Modeling Techniques; F.2.2 **[Analysis of Algorithms and Problem Complexity]**: Nonnumerical Algorithms and Problems; G.2.1 **[Discrete Mathematics]:** Combinatorics.
General Terms: Algorithms, Performance.

Keywords: Worst-Case Backoff Performance, Exponential Backoff, Polynomial Backoff, Batch, On-line.

1. 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 in a time slot, it retries in subsequent time slots, but with a diminished probability of transmission. If all packets cooperate in using this strategy and the channel is not oversubscribed, then all packets eventually transmit without interference from other packets.

Exponential backoff is well known in the context of the Ethernet [31] 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.

Exponential 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 [4, 6], TCP congestion control [34], Sun RPC congestion control [41], HTTP congestion control [7], DHCP retry [42], setting power levels on radio transmitters [44], barrier synchronization in shared-memory multiprocessors [2], optical switching [9, 11, 14], contention resolution in PRAMs [27, 30], randomized routing on fat trees [18], transaction conflict resolution in databases [40] and distributed databases [33], transactional memory access [22, 24], lock conflicts [21, 23], etc.

Given the prominent role played by exponential backoff in computer systems, it is surprising that many aspects of backoff are not yet understood. Backoff strategies have typically been analyzed by making statistical assumptions on the distribution of problem inputs [12, 13, 16, 20, 35]. Although these analyses provide valuable

insights into the efficacy of exponential backoff and its variants, it is dangerous to neglect the worst case by assuming statistical arrivals because bursty and other pathological inputs are often the common case [19, 29].

The Model

In this paper we study the worst-case performance of randomized backoff algorithms for *simple multiple-access channels* with the goal of understanding bursty packet arrivals. We begin by defining the channel model, details of exponential backoff and its generalizations, packet-arrival models, and performance measures.

In simple multiple-access channels, if several packets collide on the channel, none is successfully delivered. The only feedback to the backoff algorithm is that a transmission was not successful. Moreover, the algorithm cannot "listen" to the channel and glean information without actually attempting a transmission. (In contrast, channels where snooping is allowed are called *Ethernet channels* [30] or *full-sensing* channels [10].) 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 define and generalize exponential backoff as follows. 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 by each packet within each window[1]. A backoff strategy is *monotone* if $W_i \leq W_{i+1}$. For any constant $r > 1$, the *r-exponential* backoff algorithm is defined by $W_k = \Theta(r^k)$. When $r = 2$, we refer to the algorithm as *binary exponential backoff*. Similarly, *r-polynomial* backoff is defined by $W_k = \Theta(k^r)$. Below, we also explore a continuum of backoff strategies in which r is non-constant.

Packets are injected over time. The most common injection pattern in the literature is that the packets arrive according to a Poisson distribution with *arrival rate* λ. We study other arrival distributions in order to analyze how exponential backoff and variants behave in the presence of *bursts*. In this paper we show that the performance of backoff on Poisson arrivals differs substantially from that on bursty arrivals. We first consider the limiting case of a burst, the *batched* arrival case, in which all n jobs arrive at time 0. The problem of contention resolution with batched arrival on a simple channel is known as the *Control Tower Problem* [30] and was motivated by the routing of h-relations on optical computers.

We also consider the *adversarial queuing model* [5]. 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. Once again, λ is referred to as the arrival rate, and T is called the *interval size*. We explore the difference between Poisson arrivals and adversarial-queueing-theory arrivals. In contrast to Poisson arrivals, the adversarial model imposes an upper bound on the number of packet arrivals within an interval but allows these packets to be distributed arbitrarily. We illustrate the interplay between this strict upper bound and the burstiness within the window.

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 normalized by T. We say that a backoff strategy is (λ, T)-*unstable* if there exists a (λ, T)-stream such that the throughput is less than the arrival rate. Otherwise, a backoff strategy is (λ, T)-*stable*.

Previous Results

Most of the prior analytical results for contention resolution assume a statistical queuing-theory model, that is, Poisson packet arrivals. The literature in this area is rich (see [10] for a nice survey). We mention that Raghavan and Upfal [35] give a protocol that has $O(\log n)$ expected delay for Poisson arrivals in a model where users queue up packets that they are to send. Goldberg et al [13] give backoff strategies that achieve $O(1)$ expected response time with Poisson arrivals. Kumar and Merakos had simulation results [28] that bulk arrivals seem to lead to greater stability than Poisson, when using exponential backoff.

Several papers have bounds on polynomial backoff rather than exponential [12, 15, 20, 35]. For example, Hastad, Leighton, and Rogoff [20] show that in the model where users queue up the packets they are to send, 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.

In contrast to publication [20], we show that in the batch case, exponential backoff outperforms polynomial backoff. We believe that on bursty inputs in general, polynomial backoff backs off too slowly, and exponential backoff provides a more aggressive backoff. Batch arrivals have been considered by several authors [9, 11, 14, 16, 17, 43] with the goal of routing h-relations, involving multiple channels, while we consider a detailed analysis of one channel. The most relevant work is by Gereb-Graus and Tsantilis [9] (see also [18]), who showed that there is a backoff-backon protocol with an optimal $O(n)$ makespan for the Control Tower Problem.

There is a marked contrast between this work and that of Aldous [3]. Both papers assume the *queue-free model*, where all packets are independent. Aldous shows that, for Poisson arrivals with positive (constant) λ and for binary exponential backoff, $o(t)$ packets transmit in the first t steps. Because of the strict upper bound on the number of arrivals and despite the bursty behavior, we can achieve stability or instability depending on the arrival rate, as described below.

Our results

We begin by analyzing batched arrivals, a setting that models the limiting case of a single burst. In Section 2 we use delay-sequence arguments [37] to show that for batch arrivals, if every window has size $W = \Theta(n)$, then with high probability[2], the makespan is $n \lg \lg n \pm O(n)$.

We use this result to analyze backoff strategies with varying win-

[1] Alternatively, 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* (or *age-based*) if p_t is a function only of t, in which case all p_t are mutually independent. Although Bernoulli strategies have been extensively studied, we show that they perform substantially worse than the window protocols in this paper.

[2] 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 $\Pr\{E_p\} \geq 1 - n^{-c}$.

dow sizes. Specifically, in Section 3, we show that binary exponential backoff has makespan $\Theta(n \lg n)$ and that more generally, for any constant $r > 1$, the r-exponential backoff algorithm has makespan $\Theta(n \lg^{\lg r} n)$. We also show that for any constant $r > 1$, the r-polynomial backoff algorithm has makespan $\Theta((n/\lg n)^{1+1/r})$. Thus, both exponential and polynomial backoff are highly sensitive to constants for batched inputs.

All of these batch strategies are monotonic. In Section 4, we exhibit a monotonic backoff algorithm, which we call *loglog-iterated backoff*, that achieves makespan $\Theta(n \lg\lg n/\lg\lg\lg n)$. Thus, r-exponential backoff is not optimal for any constant r. Interestingly, loglog iterated backoff is exponential backoff with $r = 1 + o(1)$, and thus, it is superpolynomial and subexponentional. Finally, we show a lower bound for the performance of any monotonic backoff strategy. This lower bound matches the performance of loglog-iterated backoff, which is thus optimal over all monotonic backoff schemes.

In Section 5, we study adversarial packet arrivals. Our results focus on exponential backoff. We show that 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 $\lambda \geq c \lg\lg n/\lg n$. On the other hand for a sufficiently small constant c, binary exponential backoff is stable with respect to any (λ, T)-adversary, if $\lambda \leq c/\lg n$. Curiously, the loglog-iterated backoff requires smaller arrival rate λ to achieve stability than does exponential backoff.

In Section 6, we wrap up with some analysis of non-window backoff strategies and future work.

2. 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 where all windows have the same fixed size, proportional to the number of packets. We use delay-sequence arguments to prove that with high probability, all packets successfully transmit with makespan $n \lg\lg n \pm O(n)$. We use these results in Sections 3 and 4 to analyze window backoff strategies, such as exponential backoff, with asymptotically varying window sizes.

We 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 [37, 38] 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 $\langle S_1, S_2, \ldots, S_k \rangle$ 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 $S_{sum} = |S_1| + \cdots + |S_k|$, 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 [32]), we shall see in Section 3 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. $S_{\text{sum}} \geq 2(|S_1| - 1 + k - \lg|S_1|)$.

Let $\text{PROBDS}(|S_1|, S_{\text{SUM}})$ 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 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 technical novelty lies in finding a sufficiently likely set of *disjoint* delay sequences, since we cannot use the union bound.

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_{left}, u_{right}).*
4. *Let nodes v and w be the left and right children respectively of node u. Then, $u_{left} = v_{left}$ and $u_{right} = w_{left}$.*

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., $J \cap J' \neq \emptyset$.

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 that 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

$$\Pr\{(T_k, J) \text{ occurs and no other delay sequences occur}\} \geq (1-p)\left(\frac{1}{W}\right)^{2^k-1} \left(\frac{W-n}{W}\right)^{2^{k+1}-2}.$$

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

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

□

3. 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 with high probability; 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 decreases 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. This is because the number of rounds is not significantly decreased by increasing the scaling constant of the exponential backoff, while the lengths of the rounds increase significantly. 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 $\Theta(n \lg^{\lg r} n)$ 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 the window size slowly, it is not as sensitive to constants as exponential backoff. Specifically, there are $\Theta(W^{1+1/r})$ 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 and with high probability 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.

4. OPTIMAL MONOTONIC 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, even those in which different packets can back off at different rates.

The log-log iterated backoff algorithm generalizes exponential backoff in that it repeatedly doubles its window size, but it stays with each window size W for $\lg\lg W$ rounds before doubling. Alternatively, we can view the algorithm as multiplying the window by a $r = 1+\Theta(1/\lg\lg n)$ factor in each round.

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 $\lg\lg n/2$ further rounds there are fewer than $n/\ln\ln\ln n$ packets left in the system with high probability because the probability of a transmission is one over an exponential in $\Theta(\lg\lg\lg n)$, say $\Theta(1/\sqrt{\lg\lg n})$. 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 structural argument showing the existence of large core sets of synchronized windows.

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 transmit 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 without 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 a structural argument to show that while most packets' windows need not be synchronized, there exists a large core set of packets all of 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 contrast, it is known that there exists a nonmonotonic backoff/backon algorithm that achieves optimal makespan $\Theta(n)$ [9, 18]. This algorithm has window sizes that vary nonmonotonically according to a "sawtooth" pattern. 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.

5. ONLINE BACKOFF

We now turn to the *online* setting, which we analyze using an *adversarial queuing model*. Our results focus on exponential backoff and log-log iterated backoff. We show that 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 for a time that is exponential in n.

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 begin our analysis of the online case by introducing a lemma that related the probability of transmission with *contention*. The contention κ_t of the system at any time step t is the sum over all packets in the system of their probability of attempting a transmission. We note that for windowed backoff, the probabilities of attempting a run at two time steps are not positively correlated: if the attempts are within the same window, they are negatively correlated, and otherwise they are uncorrelated. This will allow us to use Chernoff bounds extensively below.

LEMMA 11. *Consider a packet j making a transmission attempt at time t when the contention in the rest of the system is κ. The probability that the transmission of this packet is successful is at most $e^{-\kappa}$ and at least $\max(0, 1-\kappa)$.*

Proof Sketch. Let the remaining packets $1, \ldots, m$ in the system have transmission probabilities $p_1, \ldots, p_m$. Packet j transmit successfully only if none of the other packets make a transmission attempt, which occurs with probability $\prod_{i \neq j}(1-p_i)$. A simple optimization argument (e.g., using Lagrange multipliers) shows that this product is maximized when all p_i are equal and minimized when the probability is concentrated on one packet. When all p_i are equal, we obtain

$$\prod_{i=1}^{m}(1-p_i) \leq (1-\kappa/m)^m \leq e^{-\kappa}.$$

When the probability is concentrated on a small number of p_i, pushing these towards 1, we obtain

$$\prod_{i=1}^{m}(1-p_i) \geq \max(0, 1-\kappa).$$

□

We are now ready to show that if $\lambda \leq c/\lg n$, for sufficiently small c, that the system remains stable for time exponential in $n \lg n$:

THEOREM 12. *There is a sufficiently small constant $c < 1$ such that for $\lambda \leq c/\lg T$, exponential backoff is stable for $e^{\Omega(n)}$ time steps, for any (λ, T)-stream.*

We prove Theorem 12 by separately bounding the contribution to the contention from packets with windows less than T and greater than T. Since a small-window packet makes $\lg T$ transmission attempts before becoming a large-window packet, we have the following lemma, whose proof is immediate:

LEMMA 13. *For exponential backoff with arrival rate $\lambda \leq c/\lg T$ and for any (λ, T)-stream, at most a c-fraction of time slots have transmission attempts of small-window packets.*

Once a packet's window has size greater than T, then each transmission attempt collides with a small-window packet with probability less than c. We show by induction that the contribution of large-window jobs to the contention is $o(1)$. Therefore each time a large-window job makes a transmission attempt, there is at least a constant probability of success, implying that on average large jobs make $O(1)$ transmission attempts and contribute at most $O(1/\lg n)$ to the contention.

We now show more formally that large window packets tolerate large arrival rates. Let *Large-Window Exponential Backoff* be the backoff strategy that has an initial window size of T and then follows exponential backoff.

LEMMA 14. *For large-window exponential backoff with arrival rate $\lambda \leq 1/4 - d$ for any constant d so that with probability at least $1 - e^{-\Omega(n)}$, the contention remains constant in each time step, implying that the system remains stable for at least an amount of time exponential in n.*

Proof. When each packet makes its ith run attempt, it has a window of size $2^{i-1}T$. We define a packet class $\mathcal{C}_i$ to be the set of all packets having window size $2^i T$. Since we are guaranteeing correctness for only an exponential amount of time, there are $O(n)$ packets classes.

We show by induction on windows that with probability at least $1 - e^{-\Omega(n)}$, there are at most n packets in each class.

This guarantee means that in a window of size T we expect at most n transmission attempts from $\mathcal{C}_0$, $n/2$ transmission attempts from $\mathcal{C}_1$, $n/4$ transmission attempts from $\mathcal{C}_2$, yielding an expected total of less than $2n$ transmission attempts and a contention of less than 2λ. An application of Chernoff bounds shows that in any window of size T, there are at most $(2+\varepsilon)$ run attempts with a probability exponentially small in both $O(n)$ and $O(\varepsilon)$.

We do the induction separately for each class $\mathcal{C}_i$, partitioning time into intervals of size $2^i T$. The base cases are immediate, since at time 0, there are no packets in the system, and in any window of size T at most n packets arrive.

We show that the expected number of packets in class i remaining at the end of a $2^i T$-length interval shrinks by a constant strictly more than half. Therefore, by an application of Chernoff bounds, with an error probability exponentially small in n, at most half the packets remain. After two of these $2^i T$-length intervals have passed, the remaining packets form the next larger class. Thus, the claim follows. □

We now conclude the proof of Theorem 12.

Proof of Theorem 12. By Lemma 13, for at least a constant fraction of time steps, no small packets make transmission attempts. Therefore, using Lemma 14 we can tolerate up to a constant arrival rate packets with large windows, and the arrival rate is only $\lambda = O(1/\lg T)$. □

We next turn to the lower bound:

THEOREM 15. *The exists a sufficiently large constant c so that exponential backoff is unstable, as long as $\lambda \geq c \lg\lg n / \lg n$.*

Proof Sketch. We exhibit a (λ, T)-stream as follows: Every $T = n/\lambda$ steps, $\Theta(n^\varepsilon)$ packets are injected ($\varepsilon \leq 1$). The remaining $n - \Theta(n^\varepsilon)$ packets are injected in a stream with one injection every $\Theta(\lg n / c \lg\lg n)$ steps. We call the first $\Theta(n^\varepsilon)$ packets the *bolus* and the following packets the *drip*.

The proof proceeds by showing that in any interval of T steps beginning with the injection of the bolus, $O(n/\text{polylog}\, n)$ packets successfully transmit. To bound the number of successful transmissions, we show that at all times the contention remains above $\Omega(\lg\lg n)$ with high probability, and thus packets complete on average every $\Omega(\text{polylog}\, n)$ steps by Lemma 11. Thus, the arrival rate of packets is larger than the transmission rate. Note that the exponent in the polylog is linear in the constant c in the arrival rate.

To bound the contention, we show that by the time the contention due to the bolus reaches $O(\lg n)$ — during which time, no packets complete with high probability — enough drip packets have been injected to contribute at least $c' \lg\lg n$ to the contention, for some $c' > 1$. Specifically, after only $n^{\varepsilon'}$ drip packets have been injected, the contention from these packets is

$$\Theta\left(\sum_{i=1}^{n^{\varepsilon'}} \frac{\lambda}{i}\right) = \Theta\left(\sum_{i=1}^{n^{\varepsilon'}} \frac{c \lg\lg n}{i \lg n}\right) = \Theta(\varepsilon' c \lg\lg n).$$

We now show by induction that for the rest of the round, the contention due to drip packets stays above $c' \lg\lg n$. Thus, most of the packets that arrive stay in the system and that these packets are evenly distributed. We divide the packet arrivals into groups of size $\Theta(\lg n)$. The probability of a successful transmission is $1/\text{polylog}\, n$, and each packet makes $O(1/\log n)$ run attempts in its first T steps, implying that each packet transmits successfully with at most some constant probability p. Thus the expected contention becomes

$$\Theta\left(\sum_{i=1}^{n^{\varepsilon'}} \frac{cp\lg\lg n}{i\lg n}\right) = \Theta(\varepsilon' cp\lg\lg n).$$

By Chernoff bounds, with high probability, the contention remains $c' \lg\lg n$.

Since in each length-T interval, most packets remain in the system, the arrival rate exceeds the transmission rate. □

Note that the adversarial job stream in the proof of Theorem 15 is only slightly bursty: a burst of size $\Omega(n^{\varepsilon})$ for any $0 < \varepsilon \leq 1$ is enough to destabilize the system as long as the constant c in the arrival rate $\lambda = c\lg\lg n/\lg n$ is sufficiently large.

Curiously, many algorithms that guarantee smaller makespans in the batch setting than exponential backoff achieve stability require lower arrival rates than exponential backoff to achieve stability. For example, using a similar approach to that above, we can show that log-log iterated backoff (see Section 4) has the following online performance:

THEOREM 16. *There is a sufficiently small constant c such that for $\lambda \leq c/(\lg\lg n \lg n)$, log-log iterated backoff is stable for any (λ, T)-stream.*

THEOREM 17. *The exists a sufficiently large constant c so that log-log iterated backoff is unstable, as long as $\lambda \geq c/\lg n$.*

6. 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 possibilities 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 monotone Bernoulli backoff strategies exist that match this bound, even without knowing n. Thus, the log-log integrated 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 algorithm can achieve the same $\Theta(n)$ makespan as the optimal windowed algorithm.

Surprisingly the batch problem has an important application in its own right. In the *RFID shopping cart problem*, one has a collection of RFID tags in a shopping cart, each tag containing a unique serial number and a small computer with a few kilobytes of memory [8,26,39]. An external transmitter broadcasts a radio frequency signal to the tags, which provides enough power for each tag to run its computer for a few steps and to broadcast a message. If more than one tag broadcasts at the same time, none of the information is successfully transmitted. One previously published solution [26] takes time $O(nk)$, where there are n objects in the shopping cart, and the tags have k-bit serial numbers. Another solution [8] runs in time $O(n\log n)$ if the system can be synchronized tightly enough. Using the sawtooth algorithm, the problem runs in time $O(n)$ without the tight synchronization requirement.

In the online case, the behavior of windowed versus Bernoulli backoff is asymptotically the same. The advantage that windowed exponential backoff has over Bernoulli exponential backoff is that the probability of collision in the windowed case squares in each window, and so, once the window gets large enough, $\log\log n$ tries suffice. With Bernoulli backoff, once the probability of execution reaches $1/n$, there is a constant probability that a job does not attempt to run at all during the next n steps, which means it can take $\log n$ rounds to clear the system. In the online case, there is already a constant probability of collision with each attempt, so windowed and Bernoulli strategies have the same asymptotic performances. Similarly, r-exponential backoff has the same behavior as binary exponential backoff.

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? 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 in Ethernets. Can this information be used to improve performance? What happens if during a collision, one of the packets succeeds, as in [36]? 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 queuing theory, will lead to more stable and higher-performing computer systems.

7. REFERENCES

[1] N. Abramson. The ALOHA system — another alternative for computer communications. In *Proc. of AFIPS FJCC*, vol. 37, pp. 281–285, 1970.

[2] A. Agarwal, M. Cherian. Adaptive backoff synchronization techniques. In *ISCA'89*, pp. 396–406, 1989.

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

[4] D. J. Bernstein. qmail — an email message transfer agent. `http://cr.yp.to/qmail.html`, 1998.

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

[6] B. Costales, E. Allman. *Sendmail*. O'Reilly, third ed., 2002.

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

[8] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, second ed., 2003. E-book at books24x7.com.

[9] M. Geréb-Graus, T. Tsantilas. Efficient optical communication in parallel computers. In *SPAA'92*, pp. 41–48, 1992.

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

[11] L. A. Goldberg, M. Jerrum, T. Leighton, S. Rao. Doubly logarithmic communication algorithms for optical-communication parallel computers. *SIAM J. Comput.*, 26(4):1100–1119, 1997.

[12] L. A. Goldberg, P. D. MacKenzie. Analysis of practical backoff protocols for contention resolution with multiple servers. In *SODA'96*, pp. 554–563, 1996.

[13] L. A. Goldberg, P. D. MacKenzie, M. Paterson, A. Srinivasan. Contention resolution with constant expected delay. *J. ACM*, 47(6):1048–1096, 2000.

[14] L. A. Goldberg, Y. Matias, S. Rao. An optical simulation of shared memory. *SIAM J. Comput.*, 28(5):1829–1847, 1999.

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

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

[17] A. G. Greenberg, S. Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *J. ACM*, 32(3):589–596, 1985.

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

[19] R. Gusella. A measurement study of diskless workstation traffic on an ethernet. *IEEE Trans. on Commun.*, 38(9):1557–1568, 1990.

[20] J. Hastad, T. Leighton, B. Rogoff. Analysis of backoff protocols for multiple access channels. In *STOC'87*, pp. 241–253, 1987.

[21] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Prog. Lang. Syst.*, 15(5):745–770, 1993.

[22] M. Herlihy, J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *ISCA'93*, pp. 289–300, 1993.

[23] M. P. Herlihy, V. Luchangco, M. Moir. Obstruction-free synchronization: Double-ended queues as an example. In *ICDCS 2003*, pp. 522–529, 2003.

[24] M. P. Herlihy, V. Luchangco, M. Moir, W. M. Scherer III. Software transactional memory for dynamic-sized data structures. In *PODC 2003*, pp. 92–101, 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] A. Juels, R. L. Rivest, M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In *Conference on Computer and Communications Security*, pp. 103–111, 2003.

[27] R. M. Karp, M. Luby, F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *STOC'92*, pp. 318–326, 1992.

[28] P. Kumar, L. Merakos. Distributed control of broadcast channels with acknowledgement feedback: Stability and performance. In *CDC'84*, 1984.

[29] W. E. Leland, M. S. Taqqu, W. Willinger, D. V. Wilson. On the self-similar nature of ethernet traffic. *Comput. Commun. Rev.*, 23(4), 1993.

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

[31] R. M. Metcalfe, D. R. Boggs. Ethernet: Distributed packet switching for local computer networks. *CACM*, 19(7):395–404, 1976.

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

[33] F. M. Pittelli, H. Garcia-Molina. Reliable scheduling in a TMR database system. *ACM Trans. Comput. Syst.*, 7(1):25–60, 1989.

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

[35] P. Raghavan, E. Upfal. Stochastic contention resolution with short delays. *SIAM J. Comput.*, 28(2):709–719, 1999.

[36] R. Rajwar, J. R. Goodman. Transactional lock-free execution of lock-based programs. In *ASPLOS-X*, pp. 5–17, 2002.

[37] A. G. Ranade. How to emulate shared memory. *J. Comput. and Syst. Sciences*, 42(3):307–326, 1991.

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

[39] S. Sarma. Towards the five-cent tag. Tech Report MIT-AUTOID-WH-006, MIT Auto-ID Center, 2001. http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-006.pdf.

[40] Sleepycat Software. The Berkeley Database version 2. http://sleepycat.com, 1997.

[41] Sun Microsystems. RPC: Remote procedure call protocol specification version 2. Internet RFC 1057, 1988.

[42] Y. T'Joens, C. Hublet, P. De Schrijver. DHCP reconfiguration extension. Internet RFC 3203, 2001.

[43] D. E. Willard. Log-logarithmic protocols for resolving ethernet and semaphore conflicts. In *STOC'84*, pp. 512–521, 1984.

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