In *Proc. Int. Conf. Acous. Speech & Sig. Proc.*, (Orlando, FL), May 2002

QUEUING WITH DISTORTION-CONTROL FOR MULTIMEDIA CONTENT

Stark C. Draper and Gregory W. Wornell

Dept. EECS, Massachusetts Institute of Technology, Cambridge, MA 02139
{scd,gww}@allegro.mit.edu

ABSTRACT

There are numerous contexts in which systems need to buffer multimedia signal content. If such a buffer overflows, signal data are lost in an uncontrolled manner, which can lead to large end-to-end distortions. However, if the queued signals are distortion-tolerant, overflows can be avoided, and significant performance gains realized, by reducing the fidelity of the signals in a gradual, controlled, manner. Based on ideas of successive-approximation source coding, we design an adaptive-buffering algorithm to minimize end-to-end distortion. This algorithm performs nearly as well as a performance bound on all possible algorithms. Perhaps most importantly, the algorithm's performance remains robust across a wide range of (unpredictable) utilization rates (input rate/output rate).

1. INTRODUCTION

Consider a finite-memory queue buffering multimedia content such as audio, images, or video. If arrival rates exceed departure rates over a span of time, the queue could overflow because of memory limitations. The ensuing data loss would result in reduced end-to-end signal reconstruction fidelity. In this paper we consider a strategy that lowers the fidelity at which signals are stored in the queue in a controlled manner, freeing memory resources to avoid uncontrolled buffer overflows. This approach aims to maximize end-to-end fidelity.

These ideas are applicable in a wide range of contexts: e.g., ad-hoc sensor networks, multimedia routers, on-microchip signal distribution architectures. We exploit the availability of successive-approximation source codes. However, rather than focusing on any of the numerous specific coding schemes in the literature (e.g [1]), we instead start from the underlying basic theory [2, 3]. This enables us to develop fundamental limits on the performance of any buffering algorithm, against which specific algorithms can be tested.

In this paper we focus on a single queue as shown in Fig. 1. The input to the queue is a sequence of unquantized (or finely-quantized) source signals. The source models we use are highly simplified in order to illustrate the design trade-offs most clearly. More complex sources are considered in [4].

This work has been supported in part by NSF Grant No. CCR-0073520, by MARCO/DARPA Contract No. 2001-CT-888, and by Hewlett-Packard.

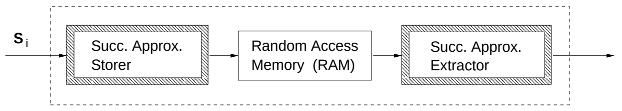

Fig. 1. Components of a queue with distortion-control.

2. INPUT AND OUTPUT DATA STREAMS

The ith signal to arrive is $\mathbf{s}_i$, and the $\mathbf{s}_i$ are each modeled as N-length sequences of independent identically distributed (i.i.d.) zero-mean Gaussian random variables of variance σ_x^2. The "normalized-quantization-rate" is defined as $R_{\mathrm{Q1}} \equiv \frac{M_{\mathrm{tot}}}{N}$, where M_{tot} is the total memory size (bits). R_{Q1} is the quantization rate (bits/sample) when all memory resources are used to describe a single observation (hence 'normalized').

The input data stream is modeled as a Poisson process of rate λ. In an interval of τ seconds, $\lambda\tau$ signals are expected. At the output, packets of M_{pac} bits are emitted according to Poisson process with rate μ_{pac}, so the average transmission rate is $M_{\mathrm{pac}}\mu_{\mathrm{pac}}$ bits/sec.

The output data stream is well described by two parameters. The first is the number of packet transmissions needed to empty the memory. This "time-to-empty" constant is defined as $\tau_{\mathrm{emp}} \equiv \frac{M_{\mathrm{tot}}}{M_{\mathrm{pac}}}$. The second is the "packet-normalized" utilization rate which measures the ratio of input to output rates: $\rho_{\mathrm{pac}} \equiv \frac{\lambda}{\mu_{\mathrm{pac}}}$.

3. SUCCESSIVE-APPROXIMATION CODES

Successive-approximation source codes are composed of K ordered subcodes, $\mathcal{C}_1, \ldots, \mathcal{C}_K$, of rates $R_1, \ldots, R_K$, respectively. Using subcodes $\mathcal{C}_1, \ldots, \mathcal{C}_l$ (where $l \leq K$), the source can be reconstructed to distortion $D(\sum_{i=1}^{l} R_i)$. Using such codes, the source can be reconstructed progressively as each subcode becomes available, rather than having to wait until the whole code is available. If a successive-approximation source code is optimal at each step, i.e., if $D(\cdot)$ is the distortion-rate function for the source, it is called a *successive-refinement* source code.

As an illustration we consider the case of the Gaussian source signals, $\mathbf{s}_i$, introduced in Section 1, under a mean-squared distortion (MSD) measure. Given any K rates, $R_1, \ldots, R_K$, using subcodes $\mathcal{C}_1, \ldots, \mathcal{C}_l$ we can describe $\mathbf{s}_i$ to within distortion $D\left(\sum_{i=1}^{l} R_i\right) = \sigma_x^2 2^{-2\sum_{i=1}^{l} R_i}$. See [2, 3, 4] for more details.

4. NON-ADAPTIVE ALGORITHM

As a baseline, we describe a non-adaptive algorithm that does not exploit ideas of successive-approximation source coding. If any part of a non-successive source code is lost, the source code becomes corrupted and therefore useless. For this reason the rates of the codes describing the signals cannot be varied dynamically. This type of algorithm is static; the designer must decide a priori at what rate to quantize each signal. As shown in [4], the optimal choice is to quantize the signals at equal rates. Given $\kappa_{\rm mem}$ is chosen as the maximum number of signals the queue can handle at any one time, each signal is quantized at rate $M_{\rm tot}/N\kappa_{\rm mem} = R_{\rm Q1}/\kappa_{\rm mem}$. At this quantization rate, the minimum achievable MSD (achievable for large N) is $E[D;\kappa_{\rm mem}] = \sigma_x^2 2^{-2\frac{R_{\rm Q1}}{\kappa_{\rm mem}}}$. Because the decoder requires the entire non-successive source code before decoding can begin, the non-adaptive algorithm sends its queued signals one at a time.

Non-Adaptive Algorithm:
0. Divide the memory into $\kappa_{\rm mem}$ blocks of size $M_{\rm tot}/\kappa_{\rm mem}$.
1. If a packet is to be sent: include all bits relevant to a particular code until the whole code has been sent.
2. If a signal is received and (a) the queue is not full then assign the signal to one of the available memory blocks. The signal is encoded at distortion $D = \sigma_x^2 2^{-2R_{\rm Q1}/\kappa_{\rm mem}}$. Otherwise (b) the queue is full. The new signal cannot be stored, is lost, and incurs distortion $D_{\max} = \sigma_x^2$.

5. ADAPTIVE ALGORITHM

The adaptive algorithm consists of two sub-algorithms with parallel structure. The first is an extraction algorithm that prioritizes sub-descriptions for inclusion in the next packet. In effect this algorithm concatenates sub-descriptions into a super-packet. The second is a storage algorithm that determines how to shuffle memory resources in order to quantize and store a newly received signal.

Suppose signals $\mathbf{s}_1, \ldots, \mathbf{s}_{m-1}$ have been received and stored. Define $k_{{\rm rec},i}$ and $k_{{\rm qu},i}$, $i = 1, \ldots, m-1$, respectively as the number of bits describing $\mathbf{s}_i$ already at the destination (i.e., already transmitted by the queue), and still contained in the queue. The description rate of each $\mathbf{s}_i$ at the destination is $R_i = k_{{\rm rec},i}/N$. Using successive-refinement source codes, the MSD at the destination is $E[D] = \frac{1}{m-1}\sum_{i=1}^{m-1}\sigma_x^2 2^{-2R_i}$. Define $\delta_{{\rm pac},i}$ to be the number of bits from the encoding of signal $\mathbf{s}_i$, to be included in the next packet. Theorem 1 gives the optimal choice for the $\delta_{{\rm pac},i}$.

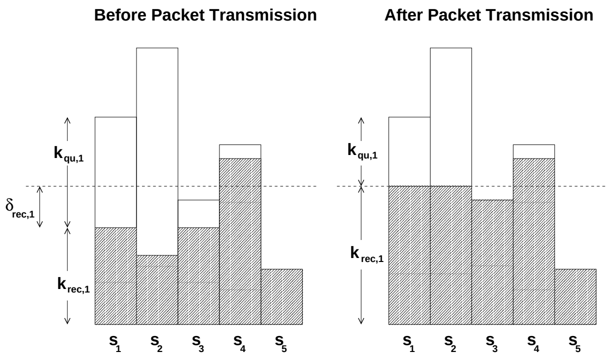
Before Packet Transmission
After Packet Transmission
$k_{\rm qu,1}$
$\delta_{\rm rec,1}$
$k_{\rm rec,1}$
$k_{\rm qu,1}$
$k_{\rm rec,1}$
s_1 s_2 s_3 s_4 s_5
s_1 s_2 s_3 s_4 s_5

Fig. 2. Determine packet contents by "water-filling" to the dashed line, which satisfies $\sum_{i=1}^{m-1}\delta_{{\rm pac},i} = M_{\rm pac}$. Shaded rectangles indicate $k_{{\rm rec},i}$ and white rectangles $k_{{\rm qu},i}$.

Theorem 1 *(Determining $\delta_{{\rm pac},i}$) Let $\mathbf{s}_i$ be as defined in Section 1. The a priori bit allocations for the $m-1$ signals at the destination and at the queue are $k_{{\rm rec},i}$ and $k_{{\rm qu},i}$, respectively. The optimal choices for the $\delta_{{\rm pac},i}$, are:*
$$\delta_{{\rm pac},i} = \min\left\{\max\left\{0, \frac{N}{2}\log\left[2\ln 2\frac{\sigma_x^2}{N}\frac{1}{\lambda}\right] - k_{{\rm rec},i}\right\}, k_{{\rm qu},i}\right\},$$
where λ is chosen so that $\sum_{i=1}^{m-1}\delta_{{\rm pac},i} = M_{\rm pac}$. The a posteriori bit allocations are $k_{{\rm rec},i}|_{\rm new} = k_{{\rm rec},i} + \delta_{{\rm pac},i}$ and $k_{{\rm qu},i}|_{\rm new} = k_{{\rm qu},i} - \delta_{{\rm pac},i}$.

This method for determining which bits are most useful to transmit is akin to "water-filling" for colored Gaussian channels in channel-coding theory, and is illustrated in Fig. 2.

Suppose $\mathbf{s}_m$ is received and must be stored. The a priori bit allocations, $k_{{\rm qu},i}$, $i = 1, \ldots, m-1$ upper-bound the a posteriori bit allocations in the queue after $\mathbf{s}_m$ has been received. Since, at most, all memory resources can be assigned to store $\mathbf{s}_m$, its a posteriori bit allocation is upper-bounded by $M_{\rm tot}$; therefore $k_{{\rm qu},m} = M_{\rm tot}$. Define $\delta_{{\rm qu},i}$ to be the changes in bit allocations made in order to store $\mathbf{s}_m$. Theorem 2 gives the optimal choice for the $\delta_{{\rm qu},i}$.

Theorem 2 *(Determining $\delta_{{\rm qu},i}$) Let $\mathbf{s}_i$ be as defined in Section 1. The a priori bit allocations for the m signals are $k_{{\rm rec},i}$ and $k_{{\rm qu},i}$. The optimal choices for the $\delta_{{\rm qu},i}$, are:* $\delta_{{\rm qu},i} =$
$$\min\left\{\max\left\{0, k_{{\rm rec},i} + k_{{\rm qu},i} - \frac{N}{2}\log\left[2\ln 2\frac{\sigma_x^2}{N}\frac{1}{\lambda}\right]\right\}, k_{{\rm qu},i}\right\}$$
where λ is chosen so that $\sum_{i=1}^{m}(k_{{\rm qu},i} - \delta_{{\rm qu},i}) = M_{\rm tot}$. The a posteriori bit allocations are $k_{{\rm rec},i}|_{\rm new} = k_{{\rm rec},i}$ and $k_{{\rm qu},i}|_{\rm new} = k_{{\rm qu},i} - \delta_{{\rm qu},i}$.

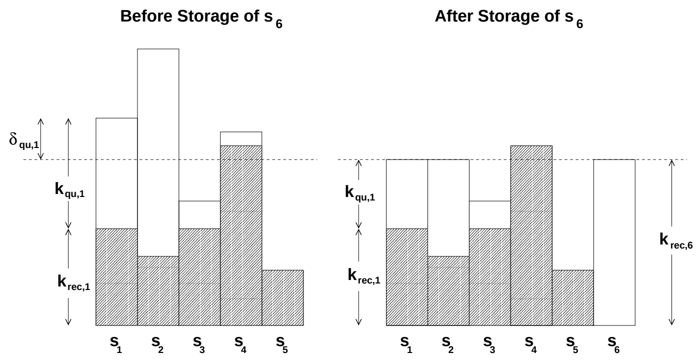

Fig. 3. Determine memory reallocations by "reverse water-pouring" to the dashed line, which satisfies $\sum_{i=1}^{m-1}(k_{\text{qu},i} - \delta_{\text{qu},i}) = M_{\text{tot}}$. Shaded rectangles indicate $k_{\text{rec},i}$ and white rectangles $k_{\text{qu},i}$.

This method for determining bit allocations is akin to "reverse water-pouring" for colored Gaussian sources in rate-distortion theory, and is illustrated in Fig. 3.

Adaptive Algorithm:

1. When deciding on the contents of the next packet for transmission: (a) Calculate $\delta_{\text{pac},i}$ for all i according to Thm. 1. (b) Concatenate the most significant $\delta_{\text{pac},i}$ bits of each signal into the next packet to be transmitted. (c) Increase $k_{\text{rec},i}$, and decrease $k_{\text{qu},i}$, by $\delta_{\text{pac},i}$.

2. When a new signal $\mathbf{s}_m$ is received, encode it using a successively-refinable source code: (a) Calculate $k_{\text{qu},i} - \delta_{\text{qu},i}$ for all i according to Thm. 2. (b) Reduce the queue memory allocated $\mathbf{s}_i$ to $k_{\text{qu},i} - \delta_{\text{qu},i}$. (c) Store $\mathbf{s}_m$ using $k_{\text{qu},m} - \delta_{\text{qu},m}$ bits.

When implementing this algorithm, there is some granularity in $\delta_{\text{pac},i}$ and $\delta_{\text{qu},i}$ that we have so far not taken into account. Basically, $\delta_{\text{pac},i}$ and $\delta_{\text{qu},i}$ must be chosen to be integer multiples of the bit-size of each of the successive-approximation sub-codes. Given there are K subcodes, theoretically $1 \ll K \ll N$. The results presented in the next section assume that K (and hence, N) is very large, so granularity effects are negligible.

6. ALGORITHMIC ANALYSIS

In [4] we derive bounds on the distortion performance of queue management algorithms. First, we show a lower bound on the normalized MSD achievable by any algorithm is

$$E\left[D\right]_{\text{norm}} = \frac{E\left[D\right]}{\sigma_x^2} \geq 2^{-2\frac{M_{\text{pac}}\mu_{\text{pac}}}{N\lambda}} = 2^{-2\frac{R_{\text{Q1}}}{\tau_{\text{emp}}\rho_{\text{pac}}}}. \quad (1)$$

Second, in [4] we show that the performance of the non-adaptive algorithm is: $E\left[D;\kappa_{\text{mem}}\right]_{\text{norm}} \geq$

$$2^{-2\frac{R_{\text{Q1}}}{\kappa_{\text{mem}}}}\left(\frac{1-\rho_{\text{sig}}^{\kappa_{\text{mem}}}}{1-\rho_{\text{sig}}^{\kappa_{\text{mem}}+1}}\right) + \frac{(1-\rho_{\text{sig}})\rho_{\text{sig}}^{\kappa_{\text{mem}}}}{1-\rho_{\text{sig}}^{\kappa_{\text{mem}}+1}}, \quad (2)$$

where ρ_{sig}, the "signal-normalized" utilization rate, is defined as $\rho_{\text{sig}} \equiv \frac{\lambda}{\mu_{\text{sig}}} = \lambda\frac{\tau_{\text{emp}}}{\kappa_{\text{mem}}\mu_{\text{pac}}}$. The reciprocal of the "signal-normalized" transmission rate, $1/\mu_{\text{sig}}$, is the average time it takes the non-adaptive queue to transmit one quantized signal. The two terms of (2) correspond to distinct sources of distortion. The first is quantization noise. This noise results from the finite memory resources available to describe each signal. The second is overflow distortion. When the queue is full, received signals are lost, and maximum distortion (σ_x^2) is incurred.

Furthermore, the performance of the non-adaptive algorithm (2) is parameterized by κ_{mem}, but is a function of ρ_{sig}. Generally, the system designer must choose κ_{mem} without knowledge of ρ_{sig}. We define $\kappa_{\text{mem,opt}}$ to be the κ_{mem} that the designer would choose given perfect knowledge of ρ_{sig}, i.e.,

$$\kappa_{\text{mem,opt}} = \arg\min_{\kappa_{\text{mem}}} E\left[D;\kappa_{\text{mem}}\right]_{\text{norm}} \simeq \rho_{\text{pac}}\tau_{\text{emp}}. \quad (3)$$

The second equality is shown to hold, to first order, in [4]. Comparison with the definition of ρ_{sig} shows that this is the κ_{mem} for which $\rho_{\text{sig}} = 1$. Substituting (3) into (2) we get a lower-bound on the union of performances across all possible non-adaptive algorithms.

Finally, simulations confirm that the performance of the adaptive memory management system always lies between the bound found by substituting (3) into (2), and the bound on all algorithms given by (1), which we rely on in the sequel.

7. COMPARISON OF ALGORITHMS

In Fig. 4 typical performance curves for the queue management algorithms are plotted versus ρ_{pac}. The lower bound on the union of performance curves of all possible non-adaptive algorithms (dashed curve), derived by substituting (3) into (2), is quite close to the bound on all algorithms (solid curve) from (1). The adaptive algorithm's performance falls between these two curves. If ρ_{pac} is known, the non-adaptive algorithm could be optimized to the particular ρ_{pac}, capitalizing on the computational simplicity of that algorithm. The disadvantage of doing this is that the performance of the non-adaptive algorithm is quite fragile and depends markedly on perfect knowledge of ρ_{pac}, as we discuss further below

The performance curves (dotted curves) of the three non-adaptive algorithms plotted in Fig. 4 have two distinct regions of operation: $\kappa_{\text{mem}} < \kappa_{\text{mem,opt}}$ and $\kappa_{\text{mem}} > \kappa_{\text{mem,opt}}$ (or $\rho_{\text{sig}} < 1$ and $\rho_{\text{sig}} > 1$). These are the 'memory-constrained' and 'communication-constrained' regions of operation, respectively. In the memory-constrained region the distortion is dominated by the first term in (2) — quantization noise — which is invariant to changes in ρ_{pac}. In the

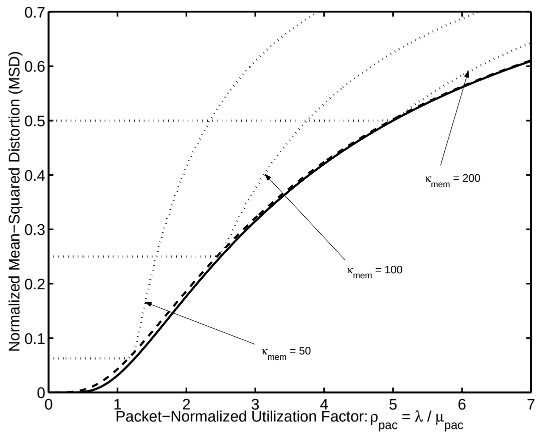

Fig. 4. Normalized MSD versus $\rho_{\rm pac}$, $R_{\rm Q1} = 100$, $\tau_{\rm emp} = 40$. MSDs for $\kappa_{\rm mem} = 50, 100, 200$ are plotted as dotted curves. These $\kappa_{\rm mem}$ are optimal for $\rho_{\rm pac} = 1.25, 2.5, 5$, respectively. The lower bounds on all non-adaptive algorithms and all algorithms are plotted as dashed and solid curves, respectively.

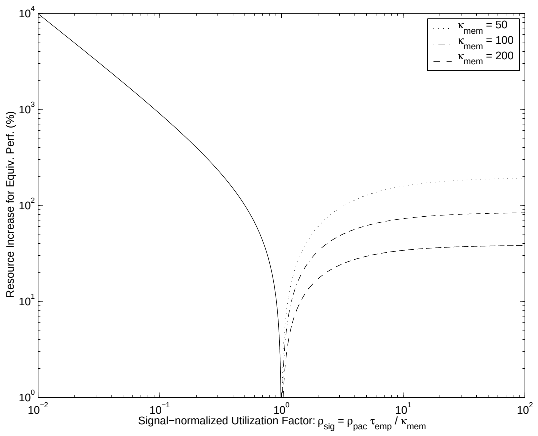

Fig. 5. The percentage by which system resources must increase so that non-adaptive algorithm performance is guaranteed to match adaptive algorithm performance for fixed $R_{\rm Q1} = 100$, $\tau_{\rm emp} = 40$, and $\kappa_{\rm mem} = 50, 100, 200$.

communication-constrained region the distortion is dominated by the second term in (2) — memory overflow — which is an increasing function of $\rho_{\rm pac}$.

To quantify the superiority of the adaptive algorithm, we determine the extra resources necessary for the performance of the non-adaptive algorithm (for $\kappa_{\rm mem}$ fixed) to match the performance bound on all algorithms (1). Matching this bound guarantees that the performance of the adaptive algorithm is also matched. Since the performance of the adaptive algorithm is quite close to the all-algorithm bound, this gives a good sense of the superiority of the adaptive algorithm. This analysis is done in [4] and the results are plotted in Fig. 5.

In the memory-constrained region, $M_{\rm tot}$ must be increase to $\gamma_{\rm mem} M_{\rm tot}$ (solid curve) for the non-adaptive performance to match the bound. In the communication-constrained region, the communication rate $\mu_{\rm pac}$ must be increase to $\gamma_{\rm com}\mu_{\rm pac}$ (dotted, dash-dotted and dashed curves) for the non-adaptive performance to match the bound. The factors $\gamma_{\rm mem}$ and $\gamma_{\rm com}$ can be shown to be: $\gamma_{\rm mem} = \frac{1}{\rho_{\rm sig}}$, and $\gamma_{\rm com} = \rho_{\rm sig}\left[\frac{1-2^{-2R_{\rm Q1}/\kappa_{\rm mem}\rho_{\rm sig}}}{1-2^{-2R_{\rm Q1}/\kappa_{\rm mem}}}\right]$. Clearly, the choice of $\kappa_{\rm mem}$ controls the performance of the non-adaptive algorithm.

If an incorrect $\kappa_{\rm mem}$ is chosen for a given $\rho_{\rm sig}$ (or $\rho_{\rm sig}$ changes or is unknown), the performance relative to the adaptive algorithm declines quickly, and significantly. This indicates that the non-adaptive algorithm is quite fragile, compared with the adaptive one. In the communication-constrained region this fragility becomes even more marked for decreasing $\kappa_{\rm mem}$. As $\kappa_{\rm mem}$ is decreased, the signal quantization rate increases, but communication resources are limited, so buffer overflows become more likely. This effect is shown for three choice of $\kappa_{\rm mem}$ (dotted, dash-dotted and dashed curves) in Fig. 5.

8. EXTENSIONS

In this paper we have assumed that the signals are quantized using a successive-refinement code, that all source signals are white, and that there are no delay constraints. We develop tools to relax the first assumption in [5], we relax the second in [4], and focus on relaxing the third in current research.

9. REFERENCES

[1] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Proc.*, vol. 41, pp. 3445–3462, Dec. 1993.

[2] W. H. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Trans. Info. Theory*, vol. 37, pp. 269–275, Mar. 1991.

[3] B. Rimoldi, "Successive refinement of information: Characterization of the achievable rates," *IEEE Trans. Info. Theory*, vol. 40, pp. 253–259, Jan. 1994.

[4] S. C. Draper and G. W. Wornell, "Distortion-controlled queuing," *In preparation*, 2001.

[5] A. Cohen, S. C. Draper, E. Martinian, and G. W. Wornell, "Stealing bits from a quantized source," *Submitted to Int. Symp. Info. Theory*, 2002.