Open access • Journal Article • DOI:10.1109/49.848236

Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction — Source link

A.E. Mohr, Eve A. Riskin, Richard E. Ladner

Institutions: University of Washington

Published on: 01 Jun 2000 - IEEE Journal on Selected Areas in Communications (IEEE)

Topics: Packet loss, Set partitioning in hierarchical trees, Image quality, Network packet and Forward error correction

Unequal Loss Protection: Graceful Degradation of Image Quality over Packet Erasure Channels Through Forward Error Correction

Alexander E. Mohr, *Student Member, IEEE*, Eve A. Riskin, *Senior Member, IEEE*, and Richard E. Ladner, *Member, IEEE*

***Abstract*—We present the unequal loss protection (ULP) framework in which unequal amounts of forward error correction are applied to progressive data to provide graceful degradation of image quality as packet losses increase. We develop a simple algorithm that can find a good assignment within the ULP framework. We use the Set Partitioning in Hierarchical Trees coder in this work, but our algorithm can protect any progressive compression scheme. In addition, we promote the use of a PMF of expected channel conditions so that our system can work with almost any model or estimate of packet losses. We find that when optimizing for an exponential packet loss model with a mean loss rate of 20% and using a total rate of 0.2 bits per pixel on the Lenna image, good image quality can be obtained even when 40% of transmitted packets are lost.**

***Index Terms*—Joint source/channel coding, lossy image transmission, lossy packet networks, packet erasure channel, packet loss, priority encoding transmission, Reed–Solomon coding, unequal loss protection.**

I. Introduction

THE INTERNET is a widely deployed network of computers that allows the exchange of data packets. In traversing the network, a packet is sent from computer to computer until it arrives at its destination. However, when the number of packets sent exceeds transmission capacity, packets are discarded at random, causing loss of data and most likely decoding failure if the lost data are not retransmitted. Each packet can be assigned a unique sequence number, so it is known which packets are received and which are lost. If the underlying transport protocol does not assign a sequence number, one or two bytes of the payload can be used to provide one. When each packet has a unique sequence number, the receiver can sort the packets according to their transmission order and any gaps in the sequence are known to be lost packets (erasures). The receiver can then take whatever action it deems best.

In networks in which packets are discarded at random, there is no way to specify the importance of a particular packet. Usually, however, the data that we transmit vary in importance. If we transmit a portrait of a face, for example, data that let us recognize the person are more important than data that show the texture of a few strands of hair. If the network is unable to transmit all of the data, then we would like it to discard the part describing the hair and retain the part that allows recognition of the face. Such a network strategy needs to quantify the importance of different chunks of data and, as channel conditions degrade, discard the least important data while retaining the most important data.

In this paper, we describe the unequal loss protection (ULP) framework that assigns unequal amounts of forward error correction (FEC) to images that are compressed with an unmodified progressive algorithm and are transmitted over lossy packet networks without using feedback. After presenting the ULP framework, we give a simple algorithm that can find a good FEC assignment within that framework. Our scheme is modular in that we can use any progressive compression algorithm and have graceful degradation of image quality with increasing packet loss rate. We focus on those packet erasure channels without feedback whose variable loss rates can be expressed as a probability mass function (PMF). Notable examples are asynchronous transfer mode (ATM) networks, wireless networks, and UDP-based transport on the Internet.

II. Background

In this section, we report previous work on protecting data from bit errors and packet losses and detail the elements that will be used in the next section to construct our ULP framework. We begin with an overview of set partitioning in hierarchical trees (SPIHT) [1] and explain ways in which it has been protected for transmission over nonideal channels. We then review the priority encoding transmission [2] scheme for using Reed–Solomon codes to protect video.

A. Set Partitioning in Hierarchical Trees

An example of a progressive image compression algorithm is SPIHT [1], an extension of Shapiro's Embedded Zerotree

Manuscript received May 5, 1999; revised November 5, 1999. This work was supported by the U.S. Army Research Office under Grant DAAH004-96-1-0255, an NSF Young Investigator Award, the NSF under Grants EIA-9973531 and CCR-9732828, and by AT&T Bell Laboratories. This paper appeared in part in the Proceedings of the Data Compression Conference, Snowbird, UT, March 29–31, 1999.

A. E. Mohr and R. E. Ladner are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350 USA (e-mail: amohr@cs.washington.edu; ladner@cs.washington.edu).

E. A. Riskin is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: riskin@ee.washington.edu).

Publisher Item Identifier S 0733-8716(00)03945-7.

Wavelet method [3]. These two new algorithms are a significant breakthrough in lossy image compression in that they give substantially higher compression ratios than prior techniques including JPEG [4], vector quantization [5], and the discrete wavelet transform [6] combined with quantization. In addition, the algorithms allow for progressive transmission [7] (meaning coarse approximations of an image can be reconstructed quickly from beginning parts of the bit stream), require no training, and are of low computational complexity.

The SPIHT algorithm uses the 9/7-tap biorthogonal filter in the discrete wavelet transform [6]. To take advantage of the self-similarity among wavelet coefficient magnitudes in different scales, the coefficients are grouped into tree structures called *zerotrees.* The organization of wavelet coefficients into a zerotree is based on relating each coefficient at a given scale (parent) to a set of four coefficients with the same orientation at the next finer scale (children). Zerotrees allow the prediction of insignificance of the coefficients across scales (that is, if the parent is insignificant with respect to a given threshold, its children are also likely to be insignificant) and represent this efficiently by coding the entire tree at once.

SPIHT groups the wavelet coefficient trees into sets and orders coefficients by the highest bit plane of the magnitude. The ordering information is encoded with a set partitioning algorithm. This algorithm is fully reproduced at the decoder. The SPIHT algorithm transmits the wavelet coefficients in bit plane order with most significant bit plane first. For each bit plane there are two passes. In the first pass, called the *dominant pass,* coefficients which are significant with respect to the current threshold are found and coded using the set partitioning method. In the second pass, the *subordinate pass,* the precision of all previously significant coefficients is increased by sending the next bit from the binary representation of their values. Such refinement allows for progressive-approximation quantization and produces a fully embedded code, i.e., the transmission of the encoded bit stream can be stopped at any point and a lower rate image can still be decompressed and reconstructed. Additionally, a target bit rate or target distortion can be met exactly.

B. Joint Source/Channel Coding Using SPIHT

Joint source/channel coding is an area that has attracted a significant amount of research effort. Despite the fact that Shannon's separation theorem [8] states that for a noisy channel, the source and channel coders can be independently designed and cascaded with the same results as given by a joint source/channel coder, complexity considerations have led numerous researchers to develop joint source/channel coding techniques. To date, most of this effort has been for fixed rate codes because they do not suffer from the synchronization problems that occur with variable rate codes [9]–[11]. (Notable exceptions that have considered joint source/channel coding schemes for variable rate codes include work on reversible variable length codes that can be decoded in both directions [12]. However, these codes can still have problems with synchronization.)

SPIHT yields high compression ratios, but images compressed with SPIHT are vulnerable to data loss. Furthermore, because SPIHT produces an embedded or progressive bit stream, meaning that the later bits in the bit stream refine earlier bits, the earlier bits are needed for the later bits to even be useful. However, SPIHT's impressive performance is leading researchers to consider transmitting images compressed with SPIHT over lossy channels and networks.

C. Prior Work on Transmitting SPIHT over Noisy Channels

Sherwood and Zeger [13] protected images compressed with SPIHT against noise from the memoryless binary symmetric channel with rate-compatible punctured convolutional (RCPC) codes [14] with good results. They extended this work to images transmitted over the Gilbert–Elliott channel (a fading channel) in [15]. In the latter case, they implement a product code of RCPC and Reed–Solomon codes and find that this outperforms the work in [13] even for the binary symmetric channel.

Rogers and Cosman were the first to consider the transmission of images compressed with SPIHT over packet erasure networks [16]. They used a fixed-length packetization scheme called packetized zerotree wavelet (PZW) compression to transmit images compressed with a modified SPIHT over lossy packet networks. The algorithm does not use any channel coding. They implemented a scheme to fit as many complete wavelet trees (i.e., one coefficient from the lowest frequency wavelet subband along with all its descendants) as possible into a packet. The algorithm degrades gracefully in the presence of packet loss because the packets are independent. If a packet is lost, they attempt to reconstruct the lowest frequency coefficients from the missing trees of wavelet coefficients by interpolating from neighboring low frequency coefficients that have been correctly received by the decoder. To simplify their algorithm, they used fewer levels of wavelet decomposition and removed the arithmetic coder from the SPIHT algorithm. The modification of the SPIHT algorithm caused a decrease of about 1.1 dB in the PSNR for the Lenna image coded at 0.209 bits per pixel for the case of a channel without losses.

These two schemes were combined into a hybrid scheme in [17]. The authors consider the case where, in addition to packet loss, packets can arrive with bit errors in them. They use channel coding to correct bit errors and PZW to conceal packet losses. If they cannot correct all of the bit errors in a packet, they consider the packet to be erased. The hybrid scheme shows resilience to packet loss, bit errors, and error bursts. It is still based on the modified SPIHT algorithm used in [16], which does not perform as well as the original SPIHT algorithm.

In recent work, Chande and Farvardin presented an unequal error protection algorithm for progressive transmission over bit error channels [18]. They assume that the bit stream can only be decoded up to the first uncorrectable error. They suggest maximizing the average useful source coding rate as an optimization criterion, because a longer prefix of the bit stream yields higher reconstructed image quality when decoded [18]. They use RCPC codes for bit errors. They use a dynamic programming approach to find the optimal code policy for each bit rate.

Their scheme shows gains over equal error protection of up to 0.6 dB.

D. Reed–Solomon Codes

Systematic Reed–Solomon (RS) codes can be used to generate FEC. RS codes are effective at recovering from erased symbols when the locations of the erased symbols are known. When packets either arrive perfectly intact or are completely discarded, we can consider RS codes that are optimized for erasures [19]. These maximum distance separable block codes are denoted by a pair (N, k), where N is the block length and k is the number of source symbols. When the code is systematic, the first k of the N encoded symbols are the source symbols and the remaining $N - k$ symbols are redundancy. They have the property that an (N, k) code can exactly recover the k source symbols from any size-k subset of the N total symbols. From an information theoretic standpoint, that property makes these codes optimal when exactly k symbols are received. This recovery is possible by treating the source symbols as the coefficients of a polynomial in a Galois field of size $2^8 = 256$ and evaluating it at a number of additional points, thus creating redundant data [19], [20].

E. Forward Error Correction for Packet Erasure Channels

Priority Encoding Transmission (PET) [2] is an algorithm that assigns FEC, according to priorities specified by the user, to message fragments (also specified by the user) sent over lossy packet networks. Each of these fragments is protected against packet losses by added FEC. It defines priorities as the fraction of transmitted packets that must be received to decode the message; thus a high priority is represented by a low percentage and the message fragment can be recovered if relatively few packets are received by the decoder. The receiver can recover the message fragment by interpolation from any subset of the transmitted packets, so long as it receives a fraction of packets at least as large as the priority of the message fragment. This property is a direct result of employing Reed–Solomon codes.

In the PET algorithm, each message fragment is assigned a fixed position within each packet. In Fig. 1, the first fragment M_1 and its FEC F_1 consist of the first L_1 bytes of each packet, the second fragment M_2 and its FEC F_2 consist of bytes from $(L_1 + 1)$ to (L_2) of each packet, and M_3 and F_3 consist of the remaining bytes of each packet. PET determines the value of L_i for each fragment and the total number of packets N, making the assumption that the number of fragments is much smaller than the number of bytes in each packet, and constrained by the user-specified priorities.

The PET algorithm does not specify how to choose the priorities to assign to the various message fragments: this assignment is left to the user. Leicher [21] applied PET to video compressed with MPEG and transmitted over packet loss channels. He used a simple three-class system in which M_1 was the intraframe (I) frames and had priority 60%, M_2 was the forward-only predicted (P) frames and had priority 80%, and M_3 was the forward–backward predicted (B) frames and had priority 95%. Thus, he can recover the I frames from 60% of the packets, the I and P frames from 80% of the packets, and all the data from 95% of the packets. This is diagrammed in Fig. 1.

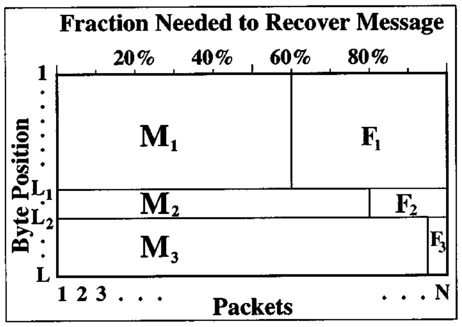


Fig. 1. In Leicher's application of PET to MPEG [21], he applied 60% priority to M_1 (I frames), 80% priority to M_2 (P frames), and 95% priority to M_3 (B frames).

Girod, Stuhlmüller, Link, and Horn applied unequal amounts of Reed–Solomon coding to protect packetized scalable H.263 video, with improved results at higher loss rates over equal or no error protection [22].

Davis *et al.* [23] presented fast lossy Internet image transmission (FLIIT) which is a joint source/channel coding algorithm. Like PET, it assigns different levels of FEC to different types of data, but it considers distortion-rate tradeoffs in its assignments. They begin with a 5-level discrete wavelet transform, create an embedded bit stream by quantizing each subband's coefficients in bit planes, apply entropy coding, and pack the bit stream from each subband into 64-byte blocks. To do bit allocation, they determine the reduction in distortion due to each block, similar to work in [24]. They then compare the greatest decrease in distortion from those blocks with the addition of a block of FEC data to the already-allocated blocks. They allocate the block of data or block of FEC that decreases the expected distortion the most. They only consider three simple cases of assigning FEC to a block: no protection, protection that consists of one FEC block shared among a group of blocks, and replication of the block. They find that, as expected, it is advantageous to apply more FEC to the coarse/low-frequency wavelet scales and to the most significant bit planes of the quantization.

The FLIIT algorithm is one of the first pieces of work to explicitly consider distortion-rate tradeoffs in making FEC assignments for lossy packet networks. However, it is limited by the coarse assignment of only three levels of protection, and the reliance on the compression algorithm they have selected (for example, SPIHT can yield a PSNR that is over 1 dB higher than their algorithm). In later work [25], the FLIIT algorithm was extended to use more powerful Reed–Solomon-like codes, but it still relies on their compression algorithm.

III. The Unequal Loss Protection Framework

While the algorithms in [15]–[17], [23] yield good results for memoryless and fading channels and for lossy packet networks, there are additional ways to transmit compressed images over lossy networks such that image quality gracefully degrades with increasing packet loss. Specifically, we will protect images transmitted over lossy channels with unequal amounts of FEC in a manner similar to the PET scheme, but we will consider the effect of each data byte on image quality when assigning protection.

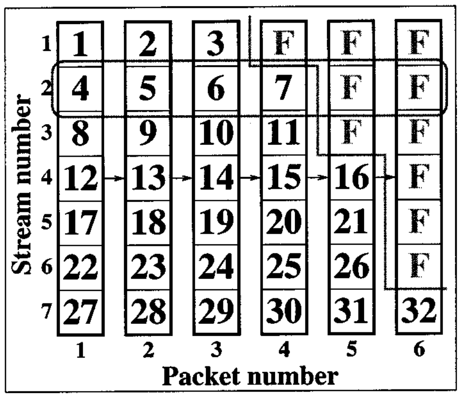
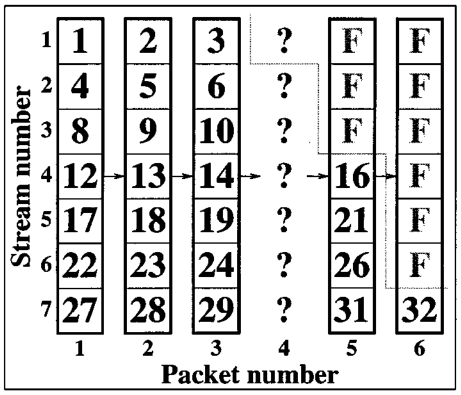


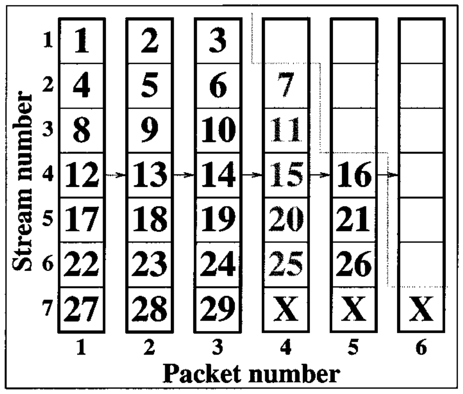
Fig. 2. Each of the rows is a stream and each of the columns is a packet. A stream contains 1 byte from each packet. The numbers 1–32 are data, and the symbol F is FEC.

In our approach to creating a framework derived from PET to assign unequal amounts of FEC to progressive data, we remove PET's restriction that the number of message fragments be much less than the number of bytes in each packet. Instead, we use a number of message fragments equal to the number of available bytes in each packet and have our algorithm dynamically choose the length and content of each message fragment. We add FEC to each message fragment to protect against packet loss such that the fragment and the FEC form a *stream.* The message is divided into L streams such that each stream has one byte of each of N packets. In Fig. 2, each of the $L = 7$ rows is a stream and each of the $N = 6$ columns is a packet. For a given stream i, for $i = 1, 2, \ldots, L$, containing both data bytes and FEC bytes, as long as the number of lost packets is less than or equal to the number of FEC bytes, the entire stream can be decoded [2]. Fig. 2 shows one possible way to send a message of 32 bytes of data (numbers 1–32) and ten bytes of FEC (F). Notice that in the figure, more bytes of FEC are applied to the earlier parts of the message and fewer are used for the later parts of the message. For SPIHT's embedded bit stream, the earlier parts of the message should have the highest priority because they are most important to the overall quality of the reproduction.

Fig. 3 shows the case where one packet out of six is lost, and five are received correctly. In this case, the first six streams can be recovered since they contain five or fewer data bytes. The last stream cannot be decoded since it contains six bytes of data and no FEC. We point out that bytes 27–29 from the seventh stream are useful since they were received correctly but bytes 31 and 32 are not useful without byte 30. Similarly, if two packets are lost, bytes 1–11 are guaranteed to be recovered and bytes 12–15 may or may not be recovered. In messages of practical length, however, those few extra bytes have only a small effect on image quality. Analogous to progressive transmission [7], even if severe packet loss occurred, we could recover a lower fidelity version of the image from the earlier streams that are decoded correctly. Each additional stream that is successfully decoded improves the quality of the received message, as long as all previous streams are correctly decoded.



(a)



(b)

Fig. 3. Demonstration of how much data can be recovered when one of six packets is lost. Here, stream 1 is unaffected by the loss, streams 2–6 use FEC to recover from the loss, and in stream 7, only the bytes up to the lost packet are useful to the decoder.

A. Formalizing the Framework

In this section, we introduce notation to formalize the ULP framework. Assume we have a message M, which is simply a sequence of data bytes to be transmitted. For example, this could be a still image compressed with SPIHT to 0.5 bits per pixel. If, instead of sending M, we send a prefix of M and some FEC, we can still maintain the same overall bit rate. We let m_i equal the number of data bytes assigned to stream i and let $f_i = N - m_i$ equal the number of FEC bytes assigned to stream i. We define the redundancy assignment, an L-dimensional FEC vector whose entries are the length of FEC assigned to each stream, as

$$\overline{f} = (f_1, f_2, \ldots, f_L).$$

For a given $\overline{f}$, we divide M into fragments $M_i(\overline{f})$ and define $M_i(\overline{f})$ to be the sequence of data bytes in the ith stream. That is, $M_i(\overline{f})$ includes the bytes of message M from position $\sum_{j=1}^{i-1} m_j + 1$ to position $\sum_{j=1}^{i} m_j$; $i = 2, 3, \ldots, L$, with $M_1(\overline{f})$ composed of m_1 bytes of stream 1. We denote a prefix of M containing the first j fragments for redundancy vector $\overline{f}$ as

$$M(j, \overline{f}) = M_1(\overline{f})M_2(\overline{f})\ldots M_j(\overline{f}).$$

We define the *incremental PSNR* of stream i

$$g_i(\overline{f}) = \mathrm{PSNR}[M(i, \overline{f})] - \mathrm{PSNR}[M(i-1, \overline{f})].$$

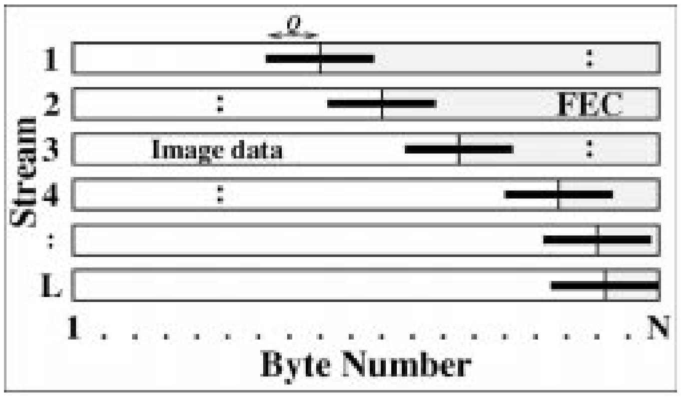


Fig. 4. At each iteration of the optimization algorithm, Q bytes of data can be added or subtracted to any of the L streams.

The quantity $g_i(\overline{f})$ is the amount by which the PSNR increases when the receiver decodes fragment i, given that all fragments prior to i have already been decoded. We set $g_1(\overline{f})$ to be the difference in PSNR between the case in which $M_1(\overline{f})$ is received and the case in which no information is received (a simple gray field).

Because the data are progressive, we require that $f_i \geq f_{i+1}$; $i = 1, 2, \ldots, L-1$; that is, the FEC assigned to the streams is nonincreasing with i. With this requirement, if $M_i(\overline{f})$ can be decoded, then $M_1(\overline{f}), M_2(\overline{f}), \ldots, M_{i-1}(\overline{f})$ can also be decoded. There is no advantage to having more redundancy in stream $i+1$ than in stream i because the loss of more than f_i packets would render both streams undecodable.

To determine the FEC vector $\overline{f}$, we use an estimate of the channel loss profile that a message is likely to encounter. This estimate is given by a PMF p_n; $n = 0, 1, \ldots, N$, such that p_n is the probability that n packets are lost. To simplify later calculations, we determine the probability that k or fewer packets are lost, and thus the cumulative distribution function is $c(k) = \sum_{n=0}^{k} p_n$; $k = 0, 1, \ldots, N$. The quantity $c(f_i)$ is the probability that receiver can decode stream i.

We can now calculate the expected PSNR of the received message as a function of $\overline{f}$ by summing over the L streams

$$G(\overline{f}) = \sum_{i=1}^{L} c(f_i) g_i(\overline{f}). \tag{1}$$

In designing an algorithm to assign FEC, we seek the $\overline{f}$ that maximizes $G(\overline{f})$ subject to a packet loss estimate p_n. Note that $g_i(\overline{f})$ could be image quality measures other than PSNR such as the mean squared error, useful source coding rate, or perceptual criteria, all of which fit within this framework.

B. Channel Estimation

In keeping with our modular design philosophy, we assume the existence of an *estimator* that outputs a PMF indicating the likelihood that a particular number of packets is lost, given the total number of packets to be transmitted. This estimator could be almost any model of expected packet loss rates: a PMF can realize uniform, binomial, Zipf, Poisson, exponential, and other distributions, as well as state-based systems such as Gilbert–Elliott channels. Furthermore, characterizing networks such as the

```
$best[*] := (N, N, \ldots, N)$
Until $best[*] = last[*]$ Do:
    $last[*] := best[*]$
    Foreach stream $s$ from 1 to L:
        Foreach $search\_value$ from $-Q$ to $+Q$:
            $temp[*] := last[*]$
            $temp[s] := temp[s] + search\_value$
            If $temp[s] < 0$ or $temp[s] > N$ then continue to next $search\_value$
            If $search\_value > 0$ then for all $i > s$
                Do $temp[i] := max(temp[s], temp[i])$
            Else for all $i < s$
                Do $temp[i] := min(temp[s], temp[i])$
            End if
            Calculate expected PSNR for $temp[*]$ using Equation 1.
            If PSNR($temp[*]$) > PSNR($best[*]$) then
                $best[*] := temp[*]$
            End if
        End foreach
    End foreach
End until
```

Fig. 5. Pseudocode of assignment algorithm. N is the number of packets, and L is the length of each packet. The variables $best$, $last$, and $temp$ are vectors that store redundancy assignments. Q is the search distance and $search\_value$ is an iteration index over that distance.

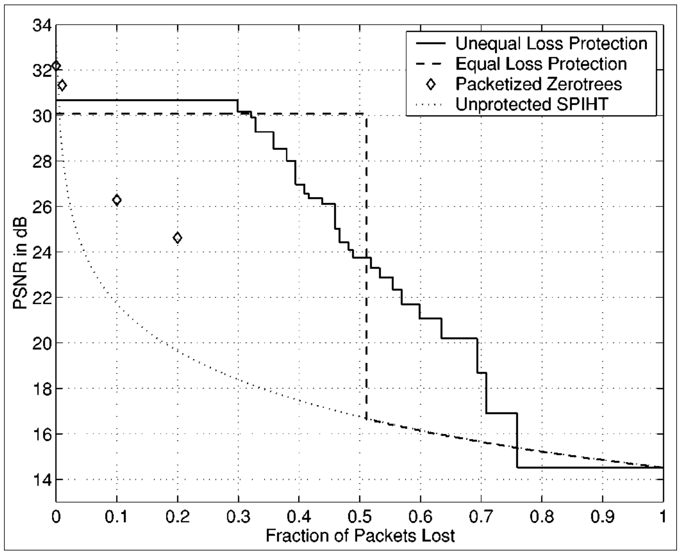


Fig. 6. Effect of packet loss on PSNR for ULP, ELP, Rogers and Cosman's packetized zerotree wavelets [16], and unprotected SPIHT. The two loss protection results are from an exponential packet loss model with a mean loss rate of 20%.

Internet is an open and active research topic in the networking community [27], [28], although we note that an estimator for the Internet is likely to be quite different from an estimator for a wireless channel. By requiring the estimator to produce a PMF, we maintain the relevance of the framework to a variety of applications and allow new developments in network channel estimation to be seamlessly incorporated into our system.

IV. An Algorithm for Solving the ULP Problem

The previous section presented a framework that can be used to assign FEC to the compressed image data. In this section, we describe an algorithm to find a good FEC assignment vector. Finding the globally optimal assignment of FEC data to each of the streams within the ULP framework appears to be computationally prohibitive for a useful amount of data.

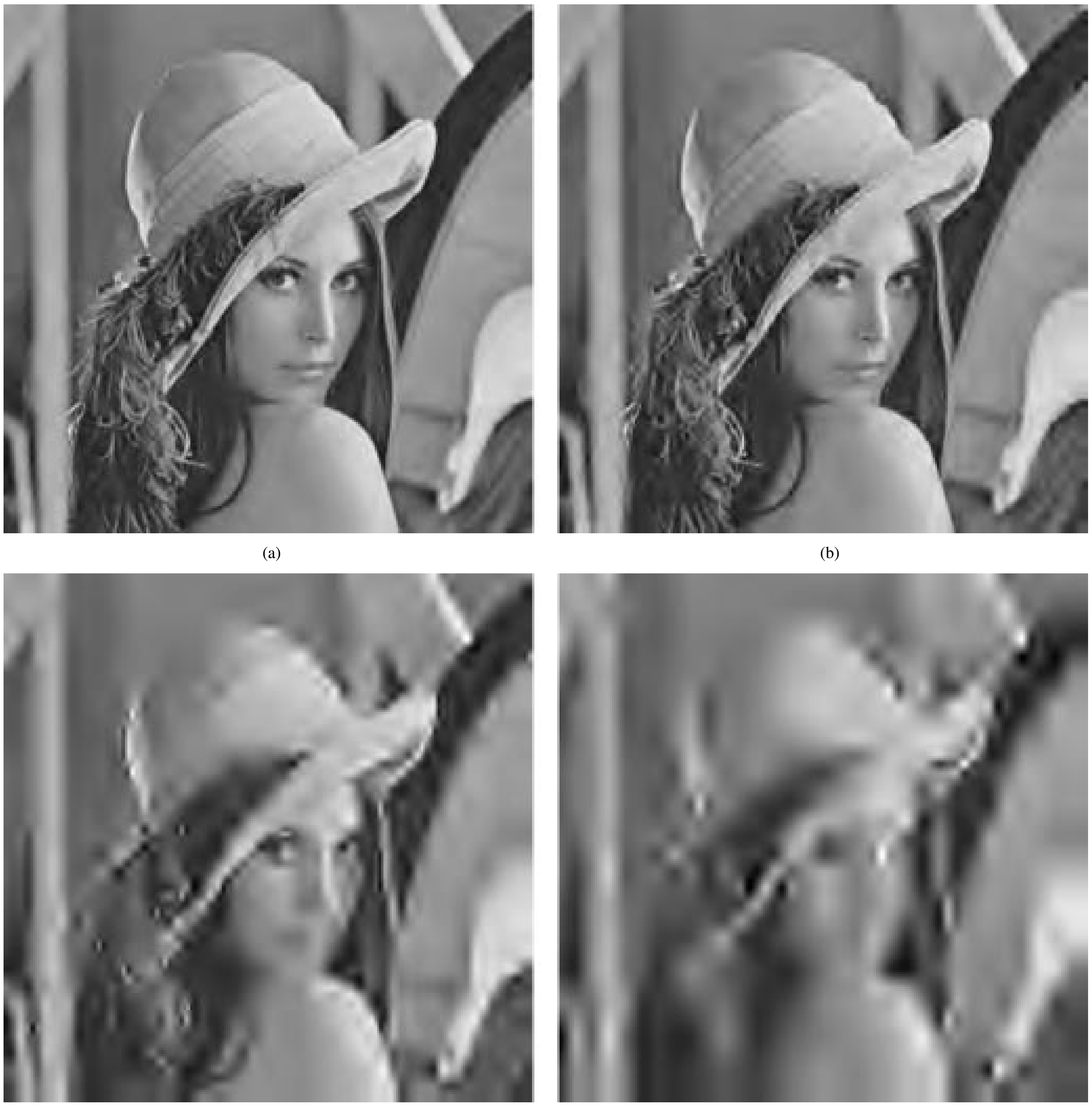


Fig. 7. Image quality at 0.2 bits per pixel total rate for Unequal Loss Protection of Lenna over a channel that has an exponential loss profile with a mean of 20%. (a) 30% of packets lost. (b) 40% of packets lost. (c) 50% of packets lost. (d) 60% of packets lost.

We therefore developed a local search hill-climbing algorithm that makes limited assumptions about the data, but is computationally tractable. As mentioned in Section III-A, we constrain $f_i \geq f_{i+1}$. Additionally, we assume that a single byte missing from the progressive bit stream causes all later bytes to become useless.

We initialize each stream to contain only data bytes, such that $m_i = N$ and $f_i = 0$; $i = 1, 2, \ldots, L$. In each iteration, our algorithm examines a number of possible assignments equal to $2QL$, where Q is the search distance (maximum number of FEC bytes that can be added or subtracted to a stream in one iteration) and L is the number of streams. We determine $G(\overline{f})$ after adding or subtracting 1 to Q bytes of FEC data to each stream (see Fig. 4), while satisfying our constraint $f_i \geq f_{i+1}$. We choose the $\overline{f}$ corresponding to the highest $G(\overline{f})$, update the allocation of FEC data to all affected streams, and repeat the search until none of the cases examined improves the expected PSNR. This process is detailed more fully in pseudocode (see Fig. 5). Our algorithm finds a local maximum that we believe is quite close to the global maximum and, in some cases, may be identical.

The search distance Q is a parameter of the algorithm that is chosen ahead of time. There is clearly a tradeoff: the larger Q is, the more likely the algorithm will find a global optimum, but the algorithm will require more time to run. When the PMF is well behaved, such as a simple unimodal function, a small Q seems to yield excellent results.

Note that for every byte of FEC data that we add to a stream, one byte of data needs to be removed. When changing the FEC assignment, we start at the first stream affected by the new allocation, move its last data byte to the next stream, move the last data byte of this stream to the following stream, and so on. This causes a cascade of data bytes to move down the streams until the last data byte from stream L is discarded. This part of the algorithm uses our assumption that the compressed sequence is progressive, because the data byte that we discard is among the least important in the embedded bit stream. The algorithm results in a set of different strength Reed–Solomon codes. The size of each code would need to be sent to the decoder as side information and how this would be implemented depends on the system being used.

V. Results

The algorithm developed in the previous section is applied to two test images. The first test image is the standard "Lenna" image and the second is a magnetic resonance image of a sagittal brain slice.

A. Lenna

For these experiments, we used the standard 512×512 gray-scale Lenna image compressed with SPIHT. We chose a total bit rate of 0.2 bits per pixel for the combination of data and FEC bytes. Because ATM packets have a payload length of 48 bytes and 1 byte is required for a sequence number, we place 47 bytes of data in each packet and send 137 packets, giving a total payload size of 6576 bytes, of which 6439 are data. Including the sequence number, the bit rate is 0.201 bits per pixel. Excluding it, the bit rate is 0.197 bits per pixel. Convergence of the algorithm is typically reached in about 27 iterations and requires 0.5 s on an Intel Pentium II 300 MHz workstation.

For this example, we use a channel loss model that is an exponential PMF with a mean loss rate of 20%. We justify the exponential shape by noting that packet loss rates are usually small, but sometimes spike to larger values. Although a 20% mean loss rate may seem high for current ATM networks, loss rates have been increasing over time [29]–[31], and such high loss rates commonly occur with wireless networks and on the Internet at peak times. We use this PMF to demonstrate that ULP is robust even in extreme situations.

We maximize the expected PSNR for two cases: ULP and equal loss protection (ELP), in which the algorithm is constrained to assign FEC equally among all of the streams. For ULP, our assignment algorithm resulted in an allocation with an expected PSNR of 29.42 dB. For ELP, the result was an allocation with an expected PSNR of 28.94 dB, or 0.48 dB lower than the ULP assignment result.

As shown in Fig. 6, under good channel conditions (packet loss rates of up to 32%, which occur 80% of the time) ULP

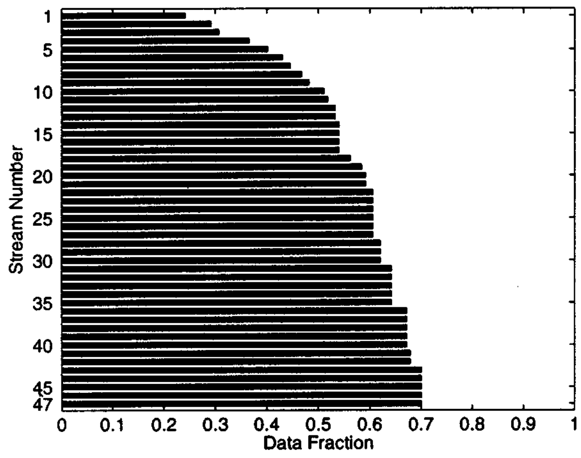


Fig. 8. Data fraction for each stream (Lenna image). Note that the FEC fraction is (1—Data Fraction). Stream 1 is the first stream (most important data), and stream 47 is the last stream (least important data).

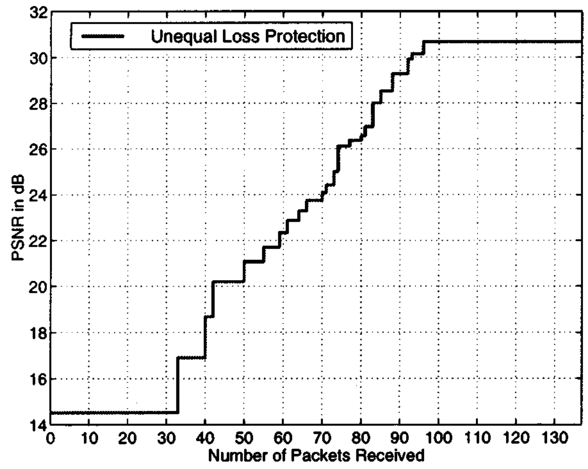


Fig. 9. The ULP system is progressive in the number of packets received (Lenna image).

yields a PSNR that is 0.66 dB higher than ELP. This is because more bytes are used for data and fewer for FEC. ELP surpasses ULP when loss rates are 33% to 51%, but those occur with only 12.5% probability. In addition, ULP degrades gracefully whereas ELP has a sharp transition at loss rates near 51%. ULP outperforms ELP a total of 85.5% of the time for this example. As expected, both of these cases substantially outperform not using any protection on the data, except when the loss rate is very low.

At those low loss rates, e.g., below 1%–2%, unprotected SPIHT will often survive with a significant prefix of the transmitted data remaining intact and the more-robust PZW coder [16] will perform slightly better. On the other hand, the performance of unprotected SPIHT and PZW degrades rapidly as losses increase, while the addition of FEC allows protected data to survive at larger loss rates. We also note that the protected data are affected only by the number of lost packets, but the reconstruction quality of unprotected SPIHT, and to a lesser extent PZW, depends upon which packets are lost. (See [26] for more discussion of this subject.)

We display results of our ULP algorithm in Fig. 7. It shows the graceful degradation of the image transmitted over a lossy

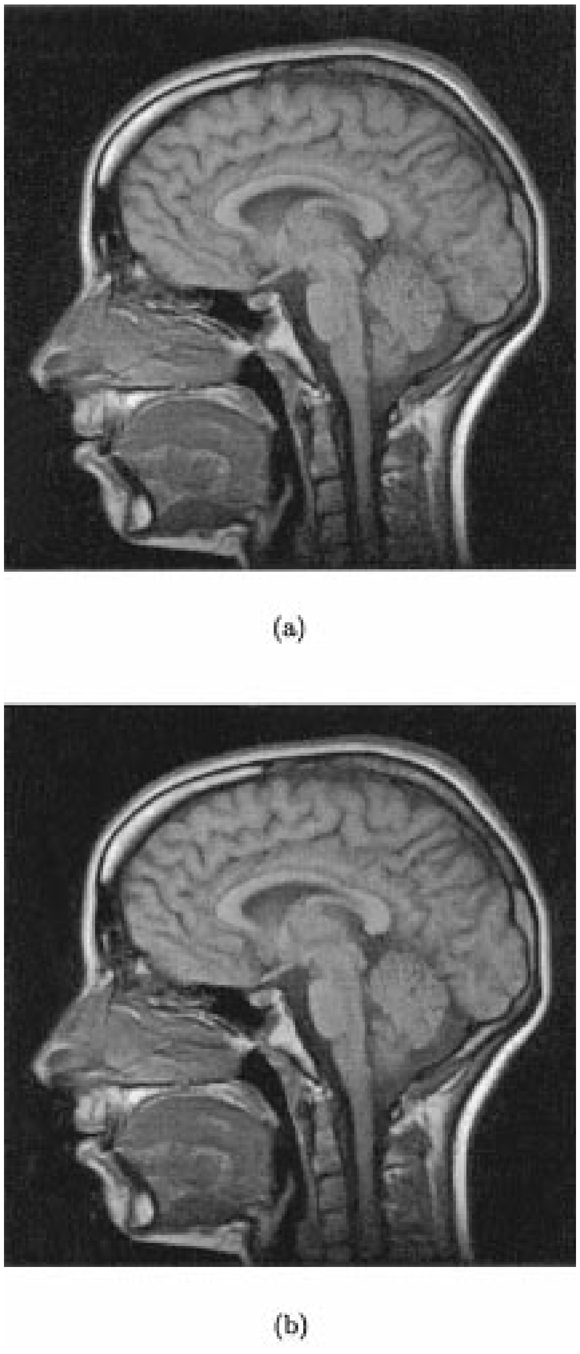


Fig. 10. The 256 × 256 magnetic resonance image. (a) The original. (b) Compressed at 1.0 bit per pixel with SPIHT.

packet network with loss rates of 30%, 40%, 50%, and 60%. Notice that the image quality remains high at a packet loss rate of 40% and the image is still recognizable at a loss rate of 50% (and even at 60% by researchers in the image compression community).

We show how ULP assigns data and FEC to the data streams in Fig. 8. Stream 1 is the first stream (most important data from the SPIHT algorithm) and it has an assignment of 24% data and 76% FEC. Stream 47 is the last stream (least important data) with 70% data and only 30% FEC. The 47 streams represent 23 different RS code strengths. As expected, the amount of FEC

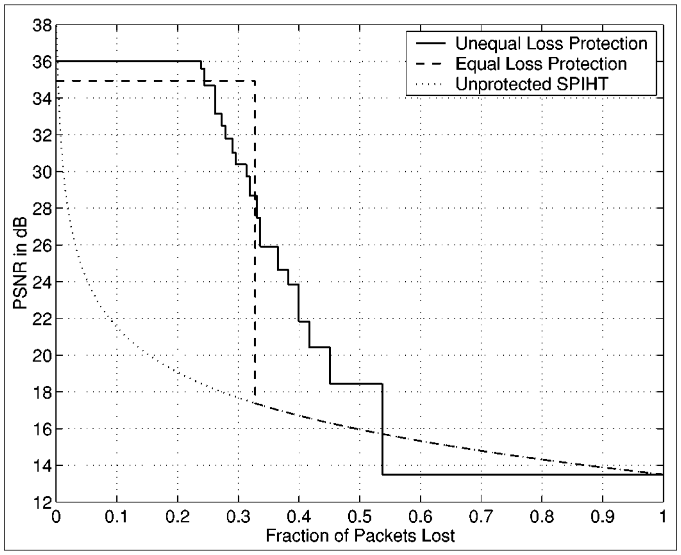


Fig. 11. Comparison of magnetic resonance image PSNR versus fraction of packets lost for ULP, ELP, and unprotected SPIHT. The channel loss model is an exponential with a mean of 10%.

decreases with increasing stream number, as required by our algorithm.

Finally, we point out that our system does provide progressive transmission, albeit with a delay. Once a number of packets equal to the number of data bytes in stream 1 is received, we can begin to decode the image. In Fig. 9, we see that when 33 packets of the ULP-protected Lenna image have arrived, stream 1 can be decoded. Then as additional packets are received, the additional decoded bytes are used to update the image. Furthermore, the image quality does not depend on which packets are received or on their order of arrival [26].

B. Magnetic Resonance Image

We next apply the ULP algorithm to a 256 × 256 magnetic resonance image of a brain compressed with SPIHT. The original image is shown in Fig. 10(a) and a compressed version at 1.0 bits per pixel is shown in Fig. 10(b). In this example, the image was transmitted in 174 47-byte payloads over a channel with an exponential mean loss rate of 10%. We chose this lower mean loss rate to demonstrate that the ULP assignment algorithm is also effective for less extreme network conditions. The total bit rate was 1.0 bits per pixel for the combination of data and FEC bytes. Convergence of the algorithm was typically reached in about 46 iterations and required 0.08 s of CPU time on an Intel Pentium II 300 MHz workstation.

In Fig. 11, we show the results of using our algorithm for both ULP and ELP. Under better channel conditions (packet loss rates of up to 24%), ULP yields a PSNR of 36.02 dB, which is 1.08 dB higher than the 34.94 dB result of ELP. As before, ULP degrades gracefully, whereas ELP would give very poor image quality if the experienced loss rate were above 33%. ULP outperforms ELP 94% of the time in this example. Fig. 12 shows the graceful degradation of the image protected with ULP and transmitted over a lossy packet network with loss rates of 10%, 20%, 30%, and 40%. Notice that the image quality remains high

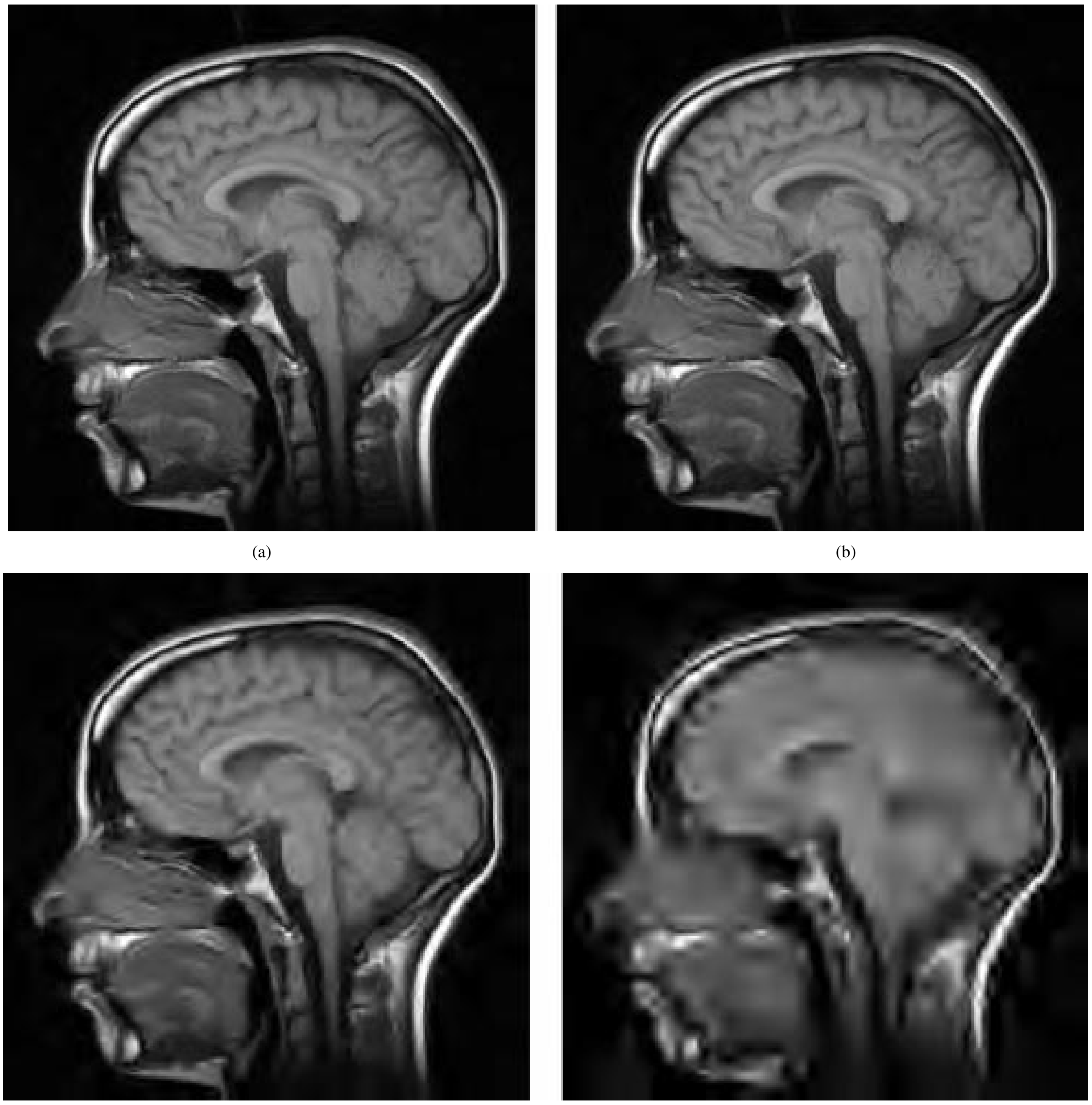


Fig. 12. Image quality at 1.0 bit per pixel total rate for Unequal Loss Protection of a magnetic resonance image over a channel that has an exponential loss profile with a mean of 10%. (a) 10% of packets lost. (b) 20% of packets lost. (c) 30% of packets lost. (d) 40% of packets lost.

at a 30% loss rate and the image is still clearly recognizable as a sagittal brain slice at the 40% loss rate.

VI. Conclusion

We have presented the Unequal Loss Protection framework and developed a simple algorithm that assigns FEC to provide graceful degradation of image quality in the presence of packet loss. Our framework is modular and can input any progressive compression scheme, any network condition estimator that produces a PMF, and other ULP assignment algorithms besides the hill-climbing algorithm presented here. As better progressive compression algorithms than SPIHT are discovered, they can be easily incorporated into the ULP framework. We are currently developing an assignment algorithm that is optimal for a convex hull approximation of the source data. We also expect to extend this work to the transmission of video sequences. Finally, we have used our ULP system to solve the generalized multiple description problem [26]. Demo programs and data files are available from http://isdl.ee.washington.edu/compression/amohr/ulp/.

ACKNOWLEDGMENT

The authors would like to thank G. W. Garrabrant for helpful discussions about lossy packet networks and Prof. W. Pearlman for the SPIHT source code.

REFERENCES

[1] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.
[2] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1737–1744, Nov. 1996.
[3] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
[4] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.
[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston: Kluwer Academic, 1992.
[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.
[7] K. Tzou, "Progressive image transmission: A review and comparison of techniques," *Opt. Eng.*, vol. 26, pp. 581–589, July 1987.
[8] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.
[9] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Trans. Inform. Theory*, vol. 36, pp. 799–809, July 1990.
[10] R.-Y. Wang, E. A. Riskin, and R. Ladner, "Codebook organization to enhance maximum *a priori* detection of progressive transmission of vector quantized images over noisy channels," *IEEE Trans. Image Processing*, vol. 5, pp. 37–48, Jan. 1996.
[11] K. Zeger and A. Gersho, "Pseudogray coding," *IEEE Trans. Commun.*, vol. 38, pp. 2147–2158, Dec. 1990.
[12] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," in *Proc. Data Compression Conf.*, Mar./Apr. 1998, pp. 471–480.
[13] P. G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Lett.*, vol. 4, pp. 189–191, July 1997.
[14] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC Codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.
[15] P. G. Sherwood and K. Zeger, "Error protection for progressive image transmission over memoryless and fading channels," *IEEE Trans. Commun.*, vol. 46, pp. 1555–1559, Dec. 1998.
[16] J. K. Rogers and P. C. Cosman, "Robust wavelet zerotree image compression with fixed-length packetization," in *Proc. Data Compression Conf.*, Mar./Apr. 1998, pp. 418–427.
[17] P. C. Cosman, J. K. Rogers, P. G. Sherwood, and K. Zeger, "Combined forward error control and packetized zerotree wavelet encoding for transmission of images over varying channels," *IEEE Trans. Image Processing*, vol. 9, June 2000.
[18] V. Chande and N. Farvardin, "Joint source-channel coding for progressive transmission of embedded source coders," in *Proc. Data Compression Conf.*, Mar. 1999, pp. 52–61.
[19] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Commun. Rev.*, vol. 27, pp. 24–36, Apr. 1997.
[20] S. Lin, J. Daniel, and J. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
[21] C. Leicher, "Hierarchical encoding of MPEG sequences using priority encoding transmission (PET)," ICSI, Tech. Rep. TR-94-058, Nov. 1994.
[22] B. Girod, K. Stuhlmüller, M. Link, and U. Horn, "Packet loss resilient Internet video streaming," in *Proc. SPIE Visual Commun. Image Processing'99*, Jan. 1999, pp. 833–844.
[23] G. M. Davis, J. M. Danskin, and X. Song, "Joint source and channel coding for Internet image transmission," in *Proc. ICIP*, vol. 1, Sept. 1996, pp. 21–24.
[24] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.
[25] G. Davis and J. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," *Proc. SPIE Appl. Digital Image Processing XIX*, pp. 376–387, Aug. 1996.
[26] A. E. Mohr, E. A. Riskin, and R. Ladner, "Generalized multiple description coding through unequal loss protection," in *Proc. ICIP*, vol. 1, Oct. 1999, pp. 411–415.
[27] S. Savage, "Sting: A TCP-based network measurement tool," in *Proc. 1999 USENIX Symp. Internet Technol. Syst.*, Oct. 1999, pp. 71–79.
[28] V. Paxson and S. Floyd, Why we don't know how to simulate the Internet, Oct. 1999. under review.
[29] D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. ACM SIGCOMM'88*, pp. 106–114.
[30] V. Paxson, "End-to-end Internet packet dynamics," in *Proc. ACM SIGCOMM*, Sept. 1997.
[31] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz, "TCP behavior of a busy Internet server: Analysis and improvements," in *Proc. IEEE Infocom Conf.*, Mar. 1998.

Alexander E. Mohr (S'00) received the B.S.E. degree in 1994 and the M.S.E. degree in 1999, both from the Department of Bioengineering, University of Washington. He is currently a Ph.D. student in the Department of Computer Science and Engineering at the same university.

His research interests have focused on the design of algorithms and systems for transmitting compressed images and video over computer communications networks.

Eve A. Riskin (SM'98) received the B.S. degree in electrical engineering from M.I.T. in 1984, and the M.S. degree in electrical engineering in 1985, the M.S. degree in operations research in 1986, and the Ph.D. degree in electrical engineering in 1990, all from Stanford University.

Since September 1990, she has been at the University of Washington where she is now an Associate Professor of Electrical Engineering. Her research interests include image compression and image processing.

Dr. Riskin is a member of Sigma Xi, Eta Kappa Nu, and Tau Beta Pi. She was awarded a National Science Foundation (NSF) Graduate Fellowship in 1984, an NSF Research Initiation Award in 1991, an NSF Young Investigator Award in 1992, and a Sloan Research Fellowship in 1994.

Richard E. Ladner (M'84) received the B.S. degree from St. Mary's College of California in 1965 and the Ph.D. degree from the University of California, Berkeley, in 1971.

Since 1971, he has been on the Faculty of the University of Washington, where he is now a Professor in the Department of Computer Science and Engineering. He has served as Editor of *SIAM Journal on Computing,* Area Editor of *Journal of the ACM,* and Associate Editor of *Journal on Computer and Systems Sciences*. His research interests include design and analysis of algorithms, computer communications and networks, and data compression.

Dr. Ladner is a Member of the IEEE Computer Society and is an ACM Fellow.