SPIHT-NC: Network-Conscious Zerotree Encoding*

Sami Iren

GTE Laboratories Incorporated
Waltham, MA 02451-1128 USA
Email: sami.iren@gte.com
Phone: (781) 466-2668 Fax: (781) 466-2130

Paul D. Amer

Computer and Information Sciences Department
University of Delaware, Newark, DE 19716 USA
Email: amer@cis.udel.edu
Phone: (302) 831-1944 Fax: (302) 831-8458

Abstract

Wavelet zerotree encoding has been proven to be an efficient way of compressing still images. Two well-known zerotree encoding algorithms, Embedded Zerotree Encoding (EZW) and Set Partitioning in Hierarchical Trees (SPIHT), provide excellent progressive display when images are transmitted over *reliable* networks. However, both algorithms are state-dependent and can perform poorly over *unreliable* networks. In this paper, we apply the concept of network-conscious image compression to the SPIHT wavelet zerotree encoding algorithm, to improve its performance over unreliable networks. Experimental results confirm the utility of network-conscious image compression concept.

1 Introduction

Wavelet zerotree encoding is an algorithm that utilizes the correlation between coefficients of different scales to provide good image compression in the wavelet domain. Wavelet zerotree encoding is based on the hypothesis that, at a given threshold level, if a wavelet coefficient at a coarse scale is insignificant, then all wavelet coefficients of the same orientation in the same spatial location at finer scales are likely to be insignificant. The embedded zerotree (EZW) encoding, originally introduced in [13], has been proven to be an efficient yet not complex encoding scheme. The embedded nature of the algorithm, a representation in which a high resolution image contains all coarser resolutions, effectively sorts bits in order of importance, thus permitting an effective progressive display when transmitted over low-bandwidth networks.

Set Partitioning in Hierarchical Trees (SPIHT), introduced in [12] as a refinement to EZW, differs from EZW in the way subsets of coefficients are partitioned and in the way significance information is conveyed. Both EZW and SPIHT provide excellent progressive

* This research is supported, in part, by the ATIRP Consortium sponsored by the US Army Research Lab under the Federal Laboratory Program (DAAL01-96-2-0002).

display. These algorithms produce a bit stream in which the bits corresponding to different trees are interleaved. When this bit stream is decoded, coefficients in all the trees are restored in parallel, thus yielding a progressive display of the whole image. What is nice about these algorithms is that they encode the most significant bit of every coefficient of every tree before encoding the next significant bit. They are designed to provide the maximum PSNR for a given significance level.

EZW and SPIHT are clearly the best *progressive display* wavelet zerotree encoding algorithms available today for *reliable* networks. For unreliable networks, however, EZW and SPIHT have a major drawback: they both are highly state-dependent, and therefore susceptible to bit errors. Even a single changed, missing, or extra bit ruins the decoding process often destroying an entire image. Recent studies concentrate on composing noise-robust zerotree encoders. Most of these studies are based on the idea of dividing the bitstream into several sub-streams each of which receive different amounts of error protection based on their noise sensitivity [10], or interleaving separately encoded substreams such that any single bit error will corrupt only one substream [4, 5].

Recently, Rogers and Cosman [11] introduced a packetized zerotree encoding (PZW) method on still images that produces fixed 53-byte ATM-compatible packets and is robust against packet loss. A similar study by Crump and Fischer [6] produced variable-length independent packets for video transmission. Sherwood and Zeger [14] improved the PZW algorithm by using a technique called “Macroscopic Multistage Image Compression”.

Unlike previous studies, which focus on robustness of the algorithm, our research on wavelet zerotree encoding is concentrated on the progressive display aspect as well as robustness, when images are transmitted over low-speed, lossy, packet-switched networks (e.g., battlefield networks). Our study of wavelet zerotree encoding is part of a broader research effort in *network-conscious image compression*[7].

A network-conscious compressed image is one that is encoded *not* simply to give the smallest size for a specified image quality, but to give the *best (i.e., smallest) response time - image quality* combination to an end user retrieving the image over a packet-switched network [9]. Network-conscious image compression is based on the concept of Application Level Framing [2]. An image is divided into path-MTU-size[1] pieces, called *Application Data Units (ADUs)*, at the application layer, so that each piece carries its semantics, that is, each piece contains enough information to be processed independently of all other ADUs. As a result, each ADU can be delivered to the receiving application immediately upon arrival at the receiver, without regard to order, thereby potentially enabling faster progressive display of images.

In this paper, we present a network-conscious version of the SPIHT algorithm called SPIHT-NC. SPIHT-NC changes the structure of the encoded bit stream to produce independent ADUs. Obviously any modification to the original structure of the SPIHT algorithm will diminish the performance of the progressive display when there is no loss as explained in Section 3. Our objective is to determine if there is any gain in making the SPIHT algorithm

[1] MTU (Maximum Transmission Unit) is the maximum frame size that a link layer can carry. A path-MTU-size ADU is one that can be transmitted end to end over a network without the need for IP layer fragmentation and reassembly.

network-conscious for faster progressive display over lossy networks. Our previous work has shown the advantages of network-conscious image compression [1, 7, 8, 9].

Section 2 explains how the network-conscious SPIHT algorithm is created. Section 3 presents experimental results and Section 4 concludes the paper with a summary of the experimental results.

2 Network-Conscious Wavelet Zerotree Encoding

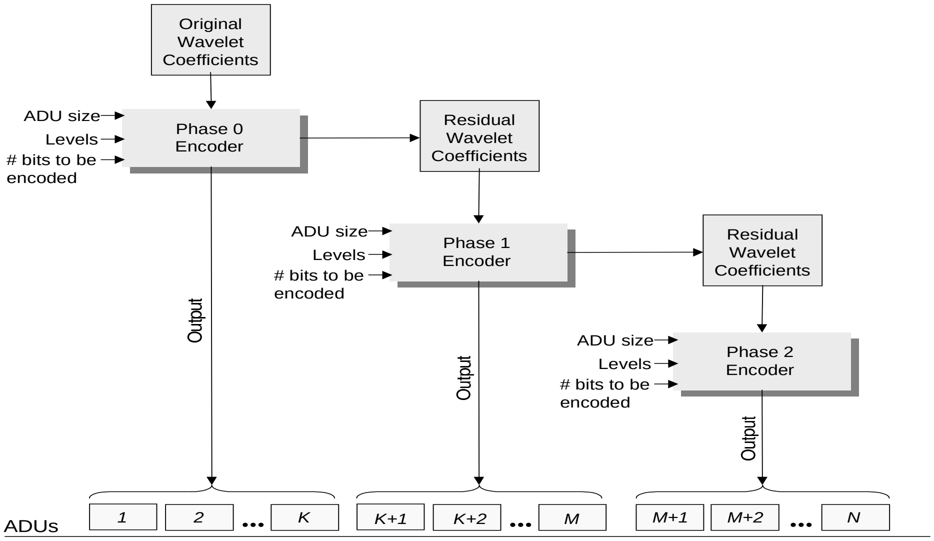
SPIHT is one of the best progressive display wavelet zerotree encoding algorithms available today. Over unreliable networks, however, SPIHT suffers from the same phenomenon every other non-network-conscious progressive compression algorithm suffers; it delays presenting out-of-order data to the user until missing pieces arrive. In this section we present SPIHT-NC, a modified version of SPIHT that performs better over unreliable networks.

To be able to create independent, SPIHT-encoded ADUs, one has to limit the state dependency to ADU boundaries. Since the smallest independent unit in zerotree encoding algorithms is a tree, it makes perfect sense to encode each tree separately rather than interleaving all the trees together. Therefore, the most primitive version of SPIHT-NC involves encoding each tree separately and packetizing these independent trees to give independent ADUs.

Roger and Cosman's PZW algorithm [11] uses this technique to introduce robustness against packet losses. This technique, called *uninterleaved spiht*, however, lacks the most important feature of network-conscious image compression: progressive display. Although some basic techniques can be applied to give the illusion of progressive display (such as making sure that trees in each packet come from widely dispersed locations in the image, and interpolating the missing coefficients from the available coefficients), our experience shows that the resulting progressive display is not satisfactory.

Among the methods we have tried to improve the progressive display of SPIHT-NC, the idea of multistage image compression proposed by Sherwood and Zeger [14] gave the best results. Our multistage technique consists of three phases (or stages) as shown in Figure 1. Each phase encodes the residual wavelet coefficients from the previous phase. Phase 0 is intended to provide a rough image with a PSNR close to what the SPIHT algorithm provides at the same bit rate. Phase 1 is optimized to refine the rough image to give an image quality that is recognizable and can be used for low bit rate applications. Phases 0 and 1 together are intended to overcome the problem of slow increase in PSNR in the early parts of the file/transmission that we observed in our earlier efforts to design SPIHT-NC. The final phase, phase 2, further refines the image obtained from phases 0 and 1 to give a high quality image.

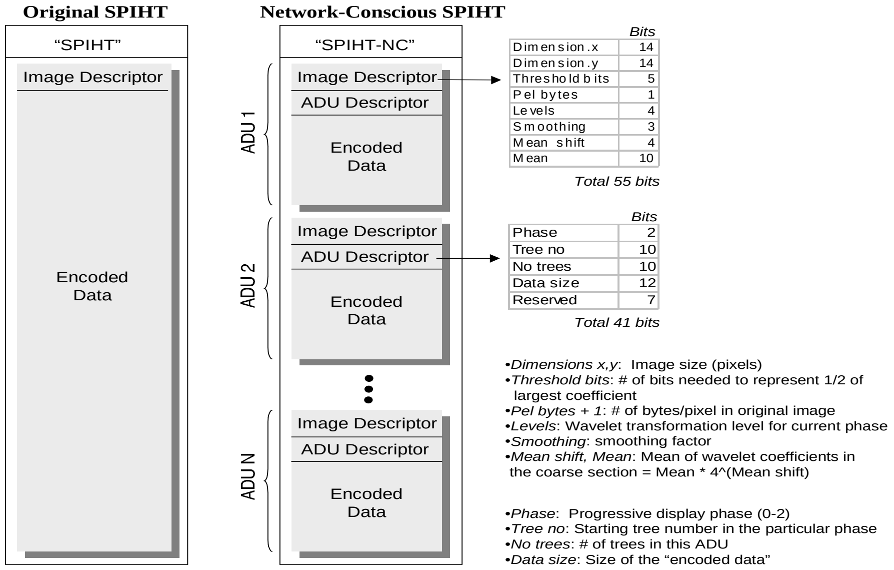
As seen in Figure 1, each phase of the encoding process inputs some parameters. These parameters are used to optimize a phase for a particular purpose. For example, for phase



[2] The number of possible wavelet decomposition levels depends on the image's dimensions.

in early ADUs. In our experiments we used the simplest method. We encoded trees at each phase in a raster scan order.

Figure 2 shows the file structures of SPIHT and SPIHT-NC. Both files start with a signature[3]. In SPIHT, an image descriptor follows that provides information about the image and some encoding parameters. The rest of the file contains the encoded data which is highly state dependent. In SPIHT-NC, after the signature, we have a sequence of ADUs each of which is self-contained. Each ADU has an image descriptor which carries image related parameters (such as dimensions of the image), an ADU descriptor which carries information specific to that particular ADU (such as phase number, starting tree number, number of trees in the ADU, etc.), and encoded data.



	Bits
Dimension.x	14
Dimension.y	14
Threshold bits	5
Pel bytes	1
Levels	4
Smoothing	3
Mean shift	4
Mean	10

Total 55 bits

	Bits
Phase	2
Tree no	10
No trees	10
Data size	12
Reserved	7

Total 41 bits

•*Dimensions x,y*: Image size (pixels)
•*Threshold bits*: # of bits needed to represent 1/2 of largest coefficient
•*Pel bytes + 1*: # of bytes/pixel in original image
•*Levels*: Wavelet transformation level for current phase
•*Smoothing*: smoothing factor
•*Mean shift, Mean*: Mean of wavelet coefficients in the coarse section = Mean * 4^(Mean shift)

•*Phase*: Progressive display phase (0-2)
•*Tree no*: Starting tree number in the particular phase
•*No trees*: # of trees in this ADU
•*Data size*: Size of the "encoded data"

[3] Said and Pearlman's original SPIHT code uses a one-byte signature (0x6E). For clarity we will use "SPIHT" for the SPIHT algorithm, and "SPIHT-NC" for the SPIHT-NC algorithm.

[4] X2E and S2E are two experimental transport protocols both developed within the Protocol Engineering Laboratory at the University of Delaware. Details of these protocols are available in [3].

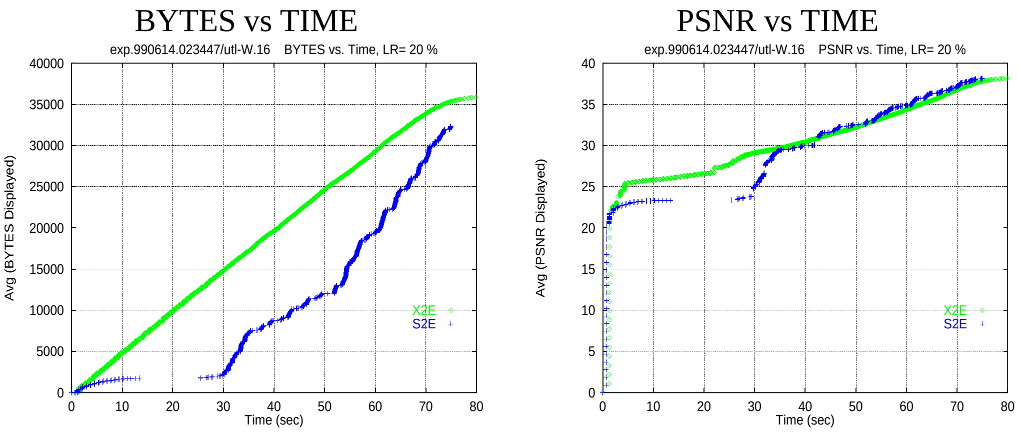
interface similar to familiar web browsers. Packets are routed through a lossy router[5] and either a PPP link or a SINCGARS radio link.

Since we would like to observe the effects of various transport and network-related parameters on the SPIHT vs. SPIHT-NC comparison, we ran several experiments that investigate parameters such as loss rate, sending transport window size, bandwidth, and transmission medium (e.g., PPP, SINCGARS, Internet). Details and results of these experiments can be found in [7]. Because of space limitations we partially present results of one of the experiments where we test the performances of SPIHT-NC vs. SPIHT at various loss rates.

3.0.1 Effect of Loss Rate

In this experiment, we investigate the progressive display advantage of SPIHT-NC vs. SPIHT at various loss rates. This experiment involves downloading the SPIHT-NC and SPIHT versions of a space shuttle image over a 9.6Kbps PPP link. Flow control between transport sender and receiver is performed via a sending window size of 16 packets. The lossy router simulates 0%, 5%, 10%, 20%, and 30% one-way IP packet loss.

Graphs presented in this paper represent averages of multiple runs for the tested parameters. For example, Figure 3 contains graphs showing the performance of SPIHT-NC (illustrated with green/gray) vs. SPIHT (illustrated with blue/black) at 20% one-way IP packet loss rate. The graph on the left shows average number of bytes delivered to the application as time progresses. The Graph on the right shows average PSNR values for the images that are displayed during the same time interval.



x

data bytes from layer k on PSNR will not be the same as the effect of x data bytes from layer l where $k \neq l$. x bytes from an upper layer (coarse scale) will result in a larger increase in PSNR than x bytes from a lower layer (finer scale).

[5] Lossy Router is an IP gateway that randomly drops certain packets according to a specified loss model and loss rate.

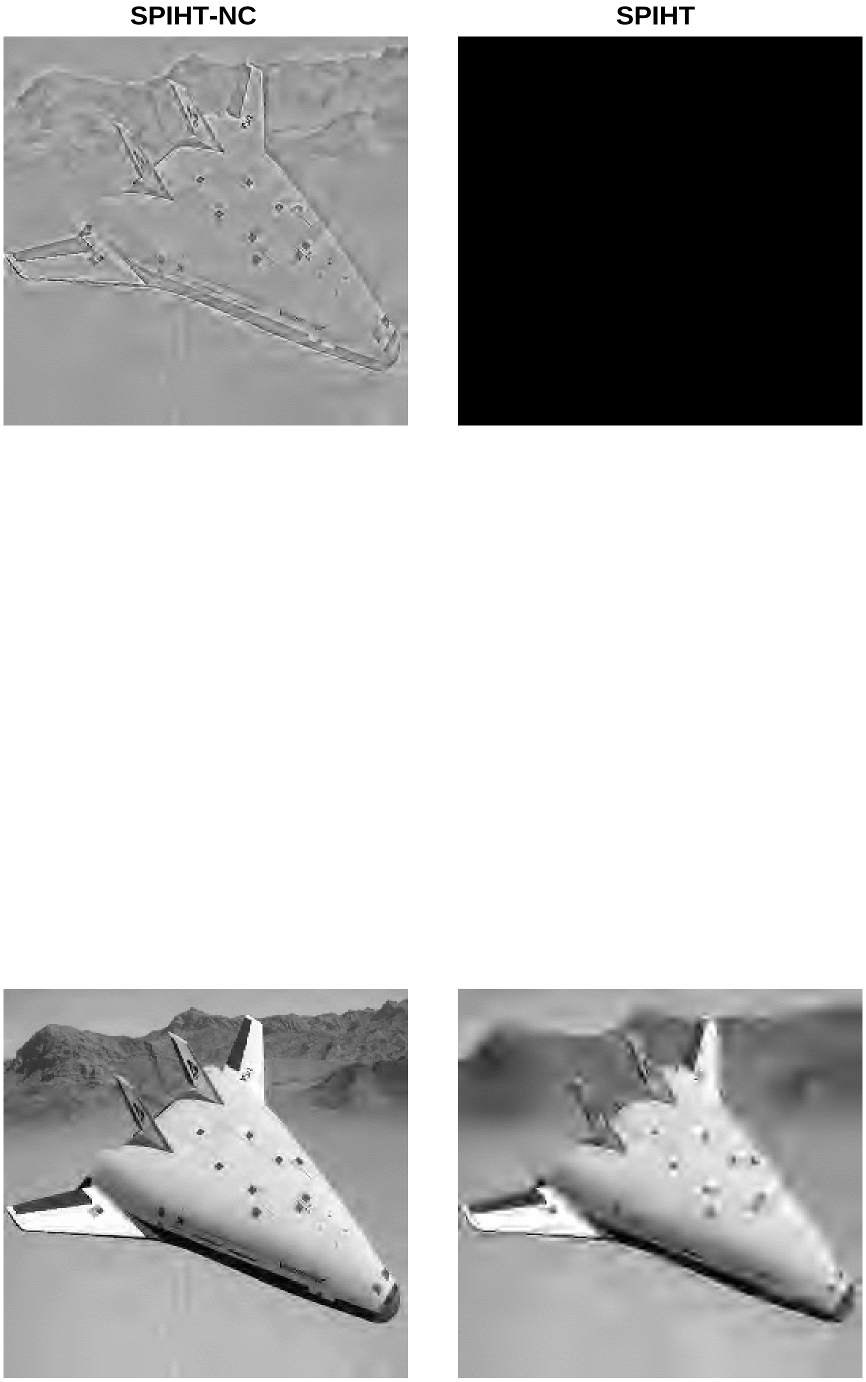
In the wavelet domain, upper layers contain significant coefficients (i.e., larger in absolute value). Since SPIHT conveys (encodes) most significant bits first, these significant coefficients from the coarse scale get transmitted first. Therefore, a sharp increase in PSNR occurs in the early moments of the transmission, and a steady increase occurs in the later part of the transmission. This sharp increase is most visible with the arrival of the first packet.

With the arrival of the first SPIHT packet, the PSNR jumps from 0 to 23.27dB. Similarly, with the arrival of the first SPIHT-NC packet, the PSNR jumps from 0 to 23.24dB. The small PSNR difference comes from the fact that SPIHT-NC has a header 5-bytes larger than SPIHT. Therefore, the first SPIHT-NC packet (ADU) carries 5-bytes less *information*. Packets 2–5 yield more modest increases in PSNR (24.96dB, 26.06dB, 26.71dB, 27.34dB with SPIHT, and 23.97dB, 24.73dB, 25.85dB, 27.15dB with SPIHT-NC). Because of the huge jump in PSNR with the first packet, it is hard to analyze the early moments of the graph. If the first packet is lost, we have a PSNR of 0dB. If the first packet is delivered to the application (i.e., not lost), we have a PSNR of more than 23dB. Considering the graphs in Figure 3 represent averages of multiple runs, the early moments of the graph will have the averages of 0's (when the first packet is lost) and 23's (when the first packet is not lost). The average of these numbers will be between 0dB and 23dB depending on the loss rate. For example, if the loss rate is 20%, out of 100 runs we expect 20 runs with 0dB (blank image) and 80 runs with 23dB.

The above statements apply to both SPIHT-NC and SPIHT. The interesting point is what happens after the first packet. With SPIHT, if the first packet is lost, the remaining packets will be buffered at the transport receiver. Therefore, no image will appear on the screen until the missing first packet is retransmitted and successfully received. With SPIHT-NC, on the other hand, even if the first packet is lost, packets that follow will be delivered to the application and an image will be displayed on the screen. Note that, since the first packet contains the coarse scale coefficients, the quality of the image produced with the remaining packets will not be as good. Nevertheless, with SPIHT-NC, at least some image will be displayed on the screen during the time that the missing first packet is retransmitted and received. In real-time applications over low-bandwidth networks, this time can be significant (i.e., life or death).

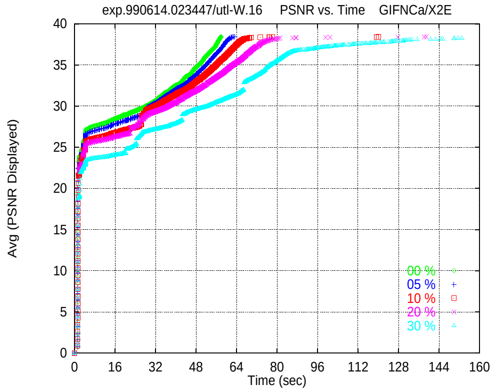
To illustrate the quality of the image in the case when packet 1 is lost, in Figure 4 we provide a sample screen shot of both SPIHT-NC and SPIHT after the first five packets are transmitted and the first packet is lost. Even though the first packet's coarse scale coefficients are lost, the coefficients in the next four packets present an image which is recognizable. In a time-critical application, SPIHT-NC will provide some information sooner than SPIHT.

Although we ran experiments at several loss rates, because of space limitations, Figure 3 shows results only at 20% loss rate. At 0% loss, SPIHT provides a better progressive display than SPIHT-NC from start to finish. Considering the excellent progressive display capability of SPIHT, this result is not surprising. One thing to note here is that SPIHT-NC performs close to SPIHT in the early moments of the transmission. Similar performance occurs because the first two phases of SPIHT-NC were optimized to perform better (i.e., close to SPIHT) at low bit rates. The performance difference is larger in the later parts of

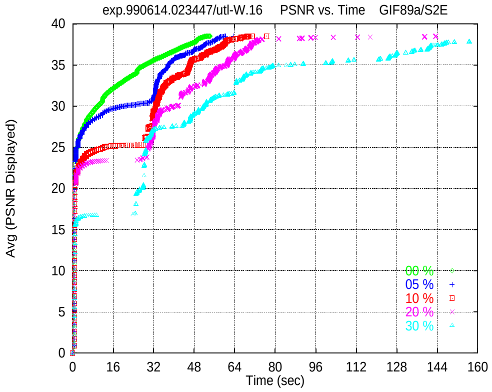

SPIHT-NC
SPIHT

play performance degrades gracefully and by smaller amounts. As the loss rate increases, the throughput of the network decreases. Therefore, as the loss rate increases, we see a shift in the graphs towards the right (i.e., it takes more time to transmit the same image at higher loss rates than lower loss rates). The shifts on the SPIHT graph occur early on resulting in poorer progressive display. The shifts on the SPIHT-NC graph are more noticeable towards the end.

SPIHT-NC



SPIHT



interactive environments where progressive display efficiency is a major consideration. Network-conscious image compression focuses not simply on maximizing compression; it focuses on optimizing overall progressive display performance.

Experimental results show that, starting at a 10% packet loss rate, SPIHT-NC outperforms SPIHT in the early moments of the progressive display where a better progressive display is more desirable. As the loss rate increases, the performance gain of SPIHT-NC over SPIHT increases due to the fact that SPIHT-NC degrades gracefully under lossy conditions. As the time it takes for a retransmission to arrive at the transport receiver increases, so does the performance gain of a network-conscious approach over a traditional approach.

References

[1] P. Amer, S. Iren, G. Sezen, P. Conrad, M. Taube, and A. Caro. Network-conscious GIF image transmission over the Internet. *Computer Networks*, 31(7):693–708, April 1999.

[2] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM '90*, pages 200–208, Philadelphia, PA, September 1990.

[3] P. Conrad. Order, reliability, and synchronization in transport layer protocols for multimedia document retrieval. PhD Dissertation, CIS Dept. University of Delaware, (in progress).

[4] C.D. Creusere. A family of image compression algorithms wich are robust to transmission errors. In *IS&T/SPIE Wavelet Applications in Signal and Image Processing IV*, volume 2825, pages 890–900, August 1996.

[5] C.D. Creusere. Image coding using parallel implementations of the embedded zerotree wavelet algorithm. In *IS&T/SPIE Symposium on Electronic Imaging*, volume 2668, San Jose, CA, 1996.

[6] V.J. Crump and T.R. Fischer. Intraframe low bitrate video coding robust to packet erasure. In *DCC '97*, Snowbird, Utah, 1997. IEEE.

[7] S. Iren. Network-conscious image compression, 1999. PhD Dissertation, CIS Dept., University of Delaware.

[8] S. Iren, P. Amer, and P. Conrad. NETCICATS: Network-conscious image compression and transmission system. *Lecture Notes in Computer Science: Advances in Multimedia Information Systems*, 1508:pp 57–68, September 1998.

[9] S. Iren, P. Amer, and P. Conrad. Network-conscious compressed images over wireless networks. *Lecture Notes in Computer Science: Interactive Distributed Multimedia Systems and Telecommunication Services*, 1483:pp 149–158, September 1998.

[10] S.H. Man and F. Kossentini. Robust EZW image coding for noisy channels. *IEEE Signal Processing Letters*, 4(8):227–229, August 1997.

[11] J. Rogers and P. Cosman. Robust wavelet zerotree image compression with fixed-length packetization. In *Data Compression Conference (DCC'98)*, Snowbird, Utah, March 1998.

[12] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3), June 1996.

[13] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Image Processing*, 41(12):3445–3462, December 1993.

[14] P. Sherwood and K. Zeger. Macroscopic multistage image compression for robust transmission over noisy channels. In *SPIE*, volume 3653, 1999.