

Resilient Packet Header Compression through Coding

Vijay Suryavanshi, Aria Nosratinia and Ramakrishna Vedantham
Multimedia Communications Laboratory, The University of Texas at Dallas
Richardson, TX 75083-0688, USA
E-mail: {vas021000, ramky, aria}@utdallas.edu

Abstract—Header compression saves bandwidth, but it also introduces error propagation whenever packets are lost. In this work we propose to use error correcting codes on the compressed packet headers. The result is an overall system that maintains most of the bandwidth savings of header compression and yet is robust with respect to errors. The key to achieving this tradeoff is appropriate distribution of parity symbols across the packets. The proposed system performs better than ordinary header compression as well as the TWICE algorithm. In most cases the effects of the error propagation can be almost removed, such that the end-to-end packet loss rate is similar to a system with no header compression, while the bitrate savings are largely maintained.

I. INTRODUCTION

Header compression techniques are necessary to reduce the packet overhead, especially when packet size is comparable to the header size. Error propagation in compressed headers is a major nuisance. The loss of a single packet (with compressed header) forces us to discard subsequent packets until the next uncompressed header is received, as shown in Figure 2.

We propose to apply forward error correction (FEC) on compressed headers. Our scheme reduces error propagation on links with no feedback. We use Reed-Solomon codes for FEC. They are efficient block codes that provide maximum loss protection for a given redundancy.

FEC encoding is not sufficient by itself to protect compressed packet headers. We need a packetization scheme that ensures the reception of enough coded symbols. Our scheme reduces the number of discarded packets by (1) protecting the compressed headers with powerful FEC and (2) uniform distribution of parity symbols among the compressed headers.

Since we apply FEC on compressed headers only (not payload), we only add moderate complexity and little redundancy. The bitrate savings obtained by our method can be used to improve the end-to-end quality of the application.

The organization of this paper is as follows. In the rest of this section, we explain the basics of header compression and summarize existing methods to stop error propagation. FEC encoding and distribution among packet headers is discussed in Section II. In Section III, we describe the channel models used in our simulations. Finally in Section IV, we present packet level simulation results and application of our scheme to video streaming.

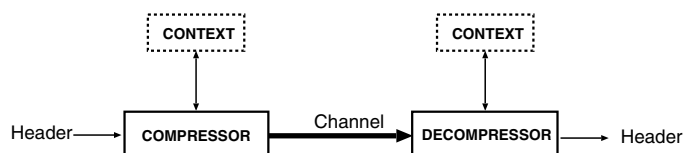


Fig. 1. Header Compression: Current headers are differentially coded from the previous header stored in the CONTEXT. After compressing each header, the CONTEXT is updated with the most recent header.

A. Header Compression Basics

Header compression schemes exploit the inter-packet correlation among the header fields. At the start of a session the transmitter sends a packet with uncompressed header followed by packets with compressed headers. As shown in Figure 1, a basic header compression configuration involves a compressor/decompressor at the transmitter/receiver and *context buffers* on both sides. One of the first header compression scheme was the Van Jacobson algorithm [1]. Although originally proposed for TCP/IP streams, this algorithm could be applied to RTP/UDP/IP¹ packets as well. RTP/UDP/IP packets are more compressible than TCP/IP packets because header fields in these packets differ by a predictable amount and these differences remain constant over a sequence of compressed packets [2] [3].

Even though very high compression is achieved, the packet discard rate is high due to error propagation. Figure 2 shows the effect of a single packet loss. As shown in Figure 2 even if one compressed packet header is lost, subsequent packets are discarded. On duplex links where feedback is possible, the decompressor can notify the transmitter that a packet has been corrupted and requests retransmission. In case of simplex links the decompressor by no means can notify the transmitter about a packet loss and has to rely on periodic updates in the form of uncompressed header packets. As shown in Figure 2 when an uncompressed header type packet arrives at the decompressor, further decompression of compressed header type packets is error free.

In header compression schemes involving TCP/IP header compression [RFC 1144], the decompressor relies on TCP retransmissions when it receives a corrupt packet. In case of

¹RTP- Real-Time Transport Protocol, UDP- User Datagram Protocol

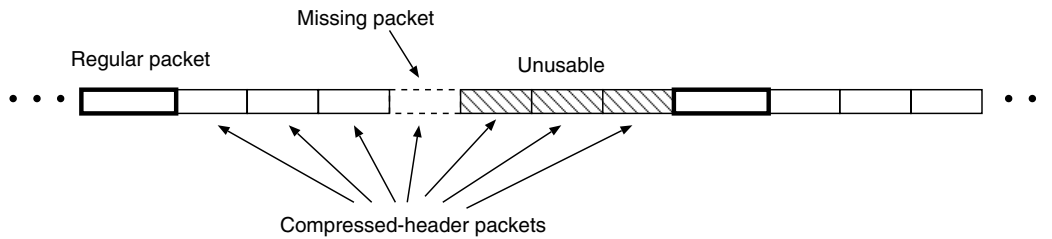


Fig. 2. Effect of Single Packet Loss

RTP/UDP/IP [RFC 2508], retransmission is not possible and it is assumed that the decompressor can notify the compressor when corrupt packets are identified. Therefore in both cases [RFC 1144 and RFC 2508] the communication is duplex. In case of simplex links where no reverse communication is possible in any form, periodic updates in form of an uncompressed header packet is transmitted.

B. Existing Schemes

One modification proposed to the Van Jacobson algorithm [1] by Perkins and Mutka [4] was to alter the way the *context buffers* were updated each time a compressed header packet was compressed/decompressed. Instead of updating the buffers at both sides each time, Perkins *et al.* proposed to update them only when an uncompressed header type packet was transmitted. This decorrelates successive header fields since each packet is compressed/decompressed according to an initial uncompressed header packet. An obvious drawback in this case is that the differences grow higher as each compressed header packet is received. Calveras *et al.* optimized this scheme [5], [6] as to when to transmit a uncompressed header packet as the differences at the tail-end grow. Of course the disadvantage in both cases is that the update can be corrupted on its way resulting in performance similar to that in [1]. A refinement to the above method was proposed by Rossi *et al.* [7], [8]. This method uses *flags* in the update packet header field and is feedback based.

A rather localized approach introduced by Degermark *et al.* was the TWICE algorithm [9]. Degermark's idea was to apply the delta values (i.e., packet header differences) *twice* to the *context buffer* assuming that the lost compressed header packet would have updated the context buffer in a similar fashion. The underlying assumption is that most of the time delta values are the same from packet to packet. The sanctity of the update (after applying TWICE) is verified by computing the checksum of the decompressed header at the transport layer (TCP or UDP). Therefore in case of RTP/UDP/IP packets, the UDP checksum disabled in most applications has to be enabled adding 2 octets (16-bit checksum) to the compressed header packet.

ECRTP (Enhanced Compressed RTP) outlined in RFC 3545 uses TWICE to recover from packet losses. The idea is to send the updates for static fields multiple times (say) N . If at-least one of these N packets are received correctly, the

context buffer can be updated correctly with TWICE providing the necessary update for the delta values. **RO**bst **C**hecksum-based **C**ompression (ROCCO) is another localized scheme which uses a 10-bit checksum computed on the original uncompressed header for reliability in case of errors. A code field indicating changes expected in the header fields is included to enable local repair of context state during desynchronization. ROCCO contributed much to the development of ROHC (**RO**bst **H**ead **C**ompression) in a way that the concept of compression profile is based on ROCCO. ROHC (RFC 3095) is specifically suited for links with high error rates and a long RTT (100-200ms).

Our proposed scheme is suited for links with no feedback and high RTT. As described in Section I-A, on these links periodic updates are necessary to refresh the *context buffer*. The packet headers of compressed header and uncompressed header packets are FEC coded and sent along with parity symbols to aid in recovering the lost compressed header packets. We discuss this in detail in the next session.

II. CODING STRATEGY

A. Forward Error Correction

Reed-Solomon (RS) codes are non-binary block codes. An (N, K) RS encoder takes K message symbols as input and produces N coded symbols. Thus it adds a redundancy of $(N - K)$ symbols to the original message. The N coded symbols are transmitted over a lossy channel. The RS decoder can recover from at-most $(N - K)$ symbol losses. i.e., it can reconstruct the K message symbols if it receives *any* K of the N transmitted symbols. A systematic RS encoder preserves the K message symbols and generates $(N - K)$ parity symbols to form a codeword of N symbols. The computational complexity is moderate for small values of N . Depending upon the symbol length m (in bits), the RS code imposes certain restriction on the codeword length N . For m -bit symbols $N \leq 2^m - 1$.

B. FEC on Packet Headers

We use $m = 8$, i.e., each symbol is a byte and hence $N \leq 255$. We apply RS code on the header symbols that we refer to as *message header bytes*. The parity symbols thus produced are referred to as *parity header bytes*.

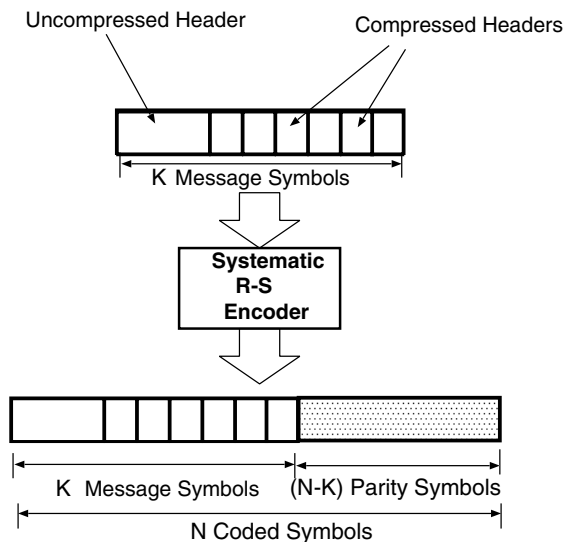


Fig. 3. Encoding the Message Header bytes. K is the message header length in bytes and N is the codeword length in bytes

The *message header bytes* (from both the uncompressed header packet and the compressed header packets) are grouped together as shown in Figure 3 and input to a systematic RS encoder.

The N coded symbols output by the RS encoder must be efficiently distributed among the packet headers so as to improve their chances of recovery. The $(N - K)$ parity header bytes are inserted² uniformly among the headers of compressed header packets as shown in Figure 4. This increases the probability that enough coded symbols are received successfully, even if some packets are lost. Once the receiver collects any K coded symbols, it can recover all the compressed packets.

We do not insert any parity header bytes in the uncompressed header because it already contains many coded symbols in the form of message header bytes. Our strategy is to equalize the number of coded symbols in each packet, so the loss of any single packet does not severely reduce the number of received coded symbols. Recall that the RS decoder requires correct reception of any K coded symbols (any combination of message or parity bytes) to recover the packet headers. Placing additional coded symbols in the uncompressed header does not serve this purpose.

In short, after encoding, the transmitted packets have the following headers:

- Uncompressed Header: The uncompressed header packet has its original *message header* bytes and therefore its header structure is the same after encoding.
- Compressed Header: The compressed header packet has its original header and at-least one parity header byte.

²We need not be particular about the placement of a parity byte. It can be the leading or trailing byte of a compressed packet header. A significant issue obviously is a general agreement between the transmitter and the receiver about the exact placement of parity symbols throughout the session.

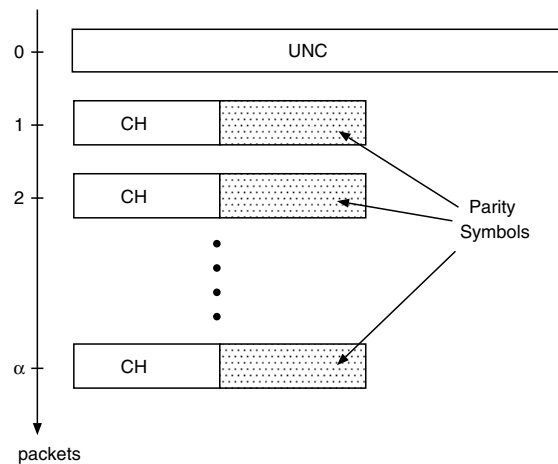


Fig. 4. Distributing the parity symbols among the compressed headers. UNC: Uncompressed Header, CH: Compressed Header

The overall restriction on N in our case can be given as:

$$N = u + \alpha c + x\alpha \leq 2^m - 1 \quad (1)$$

where

- u = uncompressed header length in symbols
- c = compressed header length in symbols
- x = number of parity symbols per compressed header packets
- α = number of compressed headers between two uncompressed headers

For a fixed value of α , various levels of error protection can be imparted to the system for different values of x . This is shown in Table I for $u = 40$, $c = 2$ and $\alpha = 40$. Note that loss of a compressed header packet means a loss of $c + x$ symbols.

C. Delay Issues

Any FEC scheme introduces some delay in encoding and decoding. In our case, the RS encoder has to wait for α packets before it can start encoding. Hence an encoding delay corresponding to α packets is incurred. This can be reduced, for example, by storing the header bytes in a buffer, RS encoding them and sending the parity symbols in the next set of α packets. For low packet loss probabilities, this won't increase the delay at the receiver, but for high packet loss rates it may double the decoder delay.

The decoder has to wait until it receives K symbols. Most of the K coded symbols are included in the uncompressed header.

TABLE I
CODE RATE FOR INCREMENTAL VALUES OF x

x	(N, K)	$c + x$
0	(120, 120)	2
1	(160, 120)	3
2	(200, 120)	4
3	(240, 120)	5

If the uncompressed header is received correctly, the decoder needs to receive a few more packets to have K symbols. Thus the decoder delay is less than α packets most of the time.

III. CHANNEL MODELS

We analyze packet loss rate of a generic simplex header compression system for two cases: i.i.d. and correlated packet losses.

1) *I.I.D. Case*: In i.i.d. case a packet is lost independently of other transmitted packets with probability p . We consider a *window* of $\alpha + 1$ packets consisting of a uncompressed header packet followed by α compressed header packets. If k denotes the number of packets discarded due to the first error event it can be easily seen that k is a random variable according to a *modified* geometric distribution as follows:

$$P(k) = (1-p)^{(\alpha+1)} \quad k = 0 \quad (2)$$

$$= p \cdot (1-p)^{(\alpha+1-k)} \quad k = 1, 2, \dots, \alpha + 1 \quad (3)$$

The average number of packets discarded can then be calculated as:

$$\bar{m} = \sum_{k=0}^{\alpha+1} k(1-p)^{(\alpha+1-k)} p = (\alpha+1) - \frac{1-p}{p} [1 - (1-p)^{\alpha+1}] \quad (4)$$

It can be shown that for a session comprising of t packets the number of uncompressed header packets is given by $N \approx \lfloor \frac{t+\alpha}{\alpha+1} \rfloor$. Therefore the total average packet loss over a given session is given as:

$$\bar{M} = \lfloor \frac{t+\alpha}{\alpha+1} \rfloor [(\alpha+1) - \frac{1-p}{p} [1 - (1-p)^{\alpha+1}]] \quad (5)$$

Thus as expected the average packet loss in a given session is a function of p and the refresh rate α .

2) *Correlated Packet Losses*: A simple and widely used model to characterize burst losses is a first-order Markov chain with 2 states '0' and '1' defined as the *good* and *bad* state respectively. A packet is received correctly if the channel is in state '0' and otherwise if in state '1'. For this 2-state Markov channel it can be shown that $P_0 = \frac{P_{10}}{P_{10}+P_{01}}$ and $P_1 = \frac{P_{01}}{P_{10}+P_{01}}$, where P_0 is the average probability of the packet being delivered correctly and timely and P_1 of being lost. Essentially P_1 is the packet loss probability. The average number of consecutive packets lost is given by $B_L = \frac{1}{P_{10}}$. As in the i.i.d. case, the average number of packets discarded in a *window* of $\alpha + 1$ packets can be calculated as:

$$\bar{m} = \sum_{k=0}^{\alpha+1} kP(k) = (\alpha+1) - P_0 [P_{01} [1 - (P_{00})^\alpha] + 1] \quad (6)$$

The i.i.d. assumption for the channel suits well in the case of Internet [10] [11]. In particular, simulations in [11] show that burst length of one (i.e., random packet errors) occur more frequently than burst losses of higher length. [10]

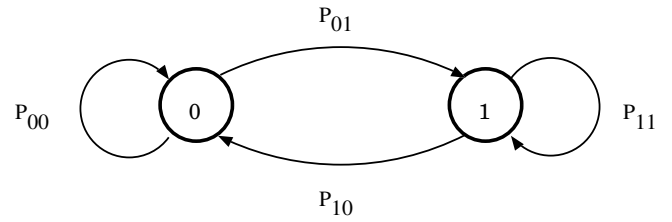


Fig. 5. A 2-state Markov Model for Correlated Packet Losses

corroborate this view with an observation that most packet losses individually occur in small number of bursts. Wireless channels on the other hand exhibit a bursty characteristic due to fading, interference and noise.

IV. SIMULATION RESULTS

A. Packet Loss Simulations

We generate random packet loss patterns according to various packet loss probabilities (1% to 30%) and burst lengths (5,10 and 15). The performance metric is the packet discard rate (PDR) after header decompression at the receiver. We employ a header compression scheme that can compress an RTP/UDP/IP packet header from 40 bytes to 3 bytes. The payload size is 500 bytes. Figure 6 compares the performance of our scheme versus the uncoded scheme, under i.i.d. packet loss conditions. Note that for a packet loss probability of 10%, our scheme is able to reduce the PDR by 70-90%. Figure 7 compares the performance under bursty packet loss conditions. In this case our scheme reduces the PDR by 50-75%. Thus our scheme significantly reduces the PDR and makes it possible to use header compression in simplex transmission. The savings in the bit rate obtained by header compression are significant, especially for low bit rate applications e.g. audio and low bit rate video transport. The bitrate savings can be used at the source coding level to improve the quality of video or audio.

B. Application to Video Transport

We consider the *Foreman* QCIF color video sequence encoded at 10 fps resulting in a 64 kbps coded bit stream. Here we compare the PSNR performance of our scheme with a scheme that employs no header compression, and a scheme that employs header compression with TWICE. Note that header compression on unidirectional links with TWICE is similar to the U/O mode of ROHC. The over-all bitrate (including header and payload) is kept constant at 64 kbps for all three cases. The same simple error concealment by temporal replenishment is used for all three cases. Figure 8 presents the PSNR performance for various packet loss probabilities and an average burst length of 5 packets. Our scheme consistently out-performs the other two schemes. TWICE performs worse than no header compression case at high packet loss rates. We observe an average PSNR gain of 2 dB compared to no header compression case, and a 3.5 dB gain over TWICE.

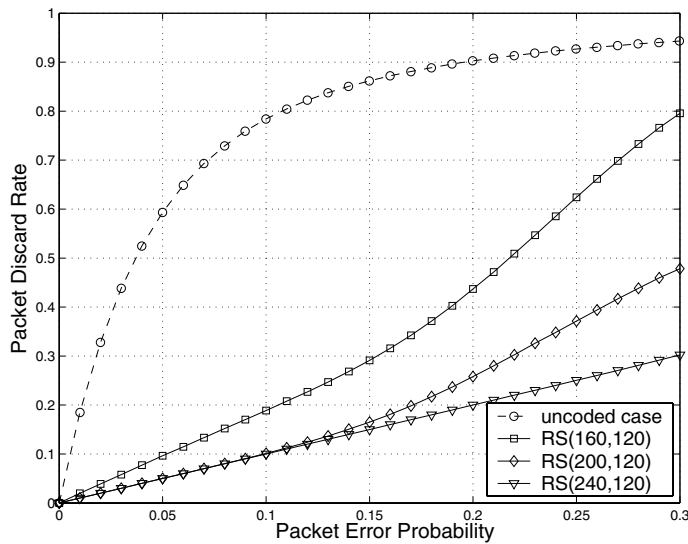


Fig. 6. I.I.D Packet Losses: Comparison of packet discard rate for the uncoded case and coded case with different levels of RS coding for $\alpha = 40$.

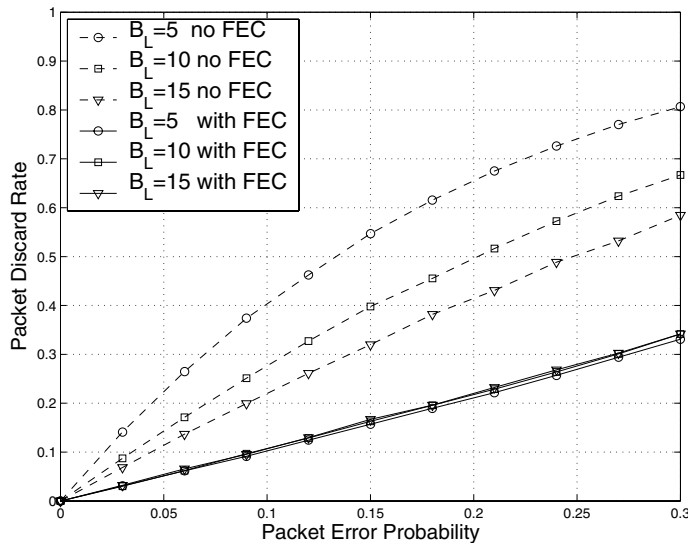


Fig. 7. Correlated Packet Losses: Comparison of packet discard rate for the uncoded case and coded case with different levels of RS coding for $\alpha = 40$, and various burst lengths (B_L)

V. CONCLUSION

We propose the use of FEC to reduce error propagation in header compression schemes with no feedback. Simulations show that our scheme significantly reduces packet discard rate at the receiver. The bitrate savings obtained by using our scheme can be transferred to source coding or channel coding or both. We transfer the bitrate savings to source coding of video and obtain a PSNR improvement of 2 dB. Our future work aims to optimally distribute the bitrate savings between the source coder and channel coder in rate-distortion optimal sense [12].

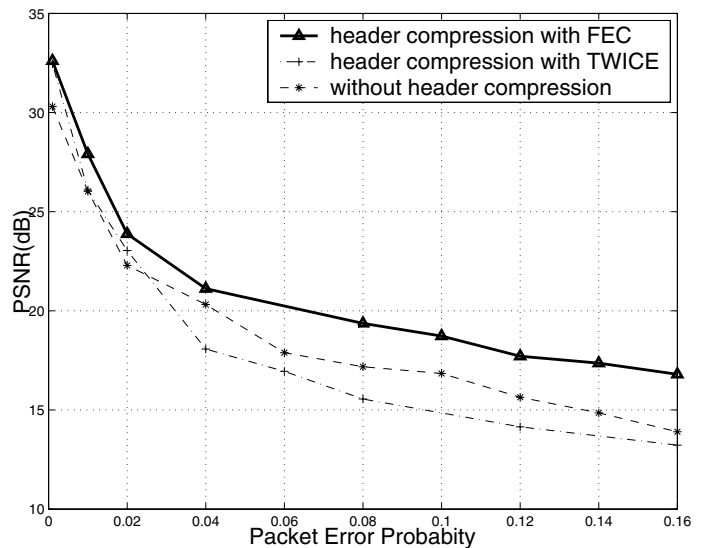


Fig. 8. PSNR comparison of the *Foreman* sequence for the three cases. Header compression with TWICE is similar to the U/O mode of ROHC

REFERENCES

- [1] V. Jacobson, "TCP/IP Compression for Low-Speed Serial Links," RFC 1144 IETF Network Working Group, Feb 1990, <http://www.ietf.org/rfc/rfc1144.txt>.
- [2] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," RFC 2508 IETF Network Working Group, Feb. 1999, <http://www.ietf.org/rfc/rfc2508.txt>.
- [3] I. Joseph, "Survey of Header Compression techniques," NASA/TM - 2001-211154, Sept 2001, National Aeronautics and Space Administration (NASA).
- [4] S. J. Perkins and M. W. Mutka, "Dependency Removal for Transport Protocol Header compression over noisy channels," *Proc. of IEEE International Conference on Communications (ICC)*, vol. 2, pp. 1025–1029, June 1997.
- [5] A. Calveras, M. Arnau, and J. Paradells, "A controlled Overhead for TCP/IP Header Compression Algorithm over Wireless Links," *Proc. of 11th International Conference on Wireless Communications (Wireless'99)*, Calgary, Canada. 1999.
- [6] —, "An Improvement of TCP/IP Header Compression Algorithm for Wireless Links," *Proc. of Third World Multiconference on Systemics, Cybernetics and Informatics (SCI 99) and the Fifth International Conference on Information Systems Analysis and Synthesis (ISAS 99)*, vol. 4, pp. 39–46, July/August. Orlando, FL. 1999.
- [7] M. Rossi, A. Giovanardi, M. Zorzi, and G. Mazzini, "Improved Header Compression for TCP/IP over Wireless Links," *Electronics Letters*, vol. 36, no. 23, pp. 1958–1960, November 2000.
- [8] —, "TCP/IP Header Compression: Proposal and Performance Investigation on a WCDMA Air Interface," *Proc. of the 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, pp. A 78–A 82, Sept 2001.
- [9] M. Degermak, M. Engan, B. Nordgren, and S. Pink, "Low loss TCP/IP header compression for Wireless Networks," in *mobicam96*, New York, NY, October 1997.
- [10] M. S. Borella, S. Uludag, G. Brewster, and D. Swider, "Internet Packet Loss: Measurement and Implications for End-to-End QoS," *International Conference on Parallel Processing Workshops*, pp. 3–12, Jan 1998.
- [11] M. Arai, A. Chiba, and K. Iwasaki, "Measurement and Modeling of Burst Packet Losses in Internet end-to-end Communications," *1999 Pacific Rim International Symposium on Dependable Computing, 1999*, pp. 260–267, Dec 1999.
- [12] A. Ortega and K. Ramchandran, "Rate-distortion Methods for Image and Video Compression," *IEEE Signal Processing Magazine*, vol. 15, pp. 23–50, November 1998.