Error-Resilient Packet Header Compression

Vijay Suryavanshi and Aria Nosratinia, *Senior Member, IEEE*

Abstract—Full packet headers consume valuable bitrate, which is especially costly in satellite links and some terrestrial wireless links. This has motivated the compression of packet headers by exploiting their correlation via using finite-state machines. The drawback is that compression in the presence of channel errors (packet loss) may result in error propagation. We offer several designs by adapting error control codes for the requirements of packet header compression in uni-directional and bi-directional links, and explore the tradeoffs in complexity, delay, and system performance. For the bi-directional link, we propose a new design called *predictive hybrid ARQ* and evaluate its performance. Experiments show significant gains in link-layer throughput as well as improved application layer performance demonstrated via video transfer experiments.

Index Terms—Header compression, ROHC, error resilience.

I. INTRODUCTION

SEVERAL studies have shown that the data traffic over the Internet is dominated by shorter packet lengths [1], [2]. These studies show that the mean as well as the median of the packet lengths is below 200 bytes. Furthermore, around 30% of packets have no payload at all, and convey only one bit of information (ACK or NACK packets), yet they must still carry a full packet header, which is 40 bytes (octets) in IPv4 and 60 bytes in IPv6. By any measure, this is not an efficient use of bandwidth.

Motivated by the arguments outlined above, several header compression algorithms have been proposed and studied over the past 15 years. The motivation for header compression and its basic feasibility is now well-accepted so that packet header compression is allowed as a part of 1xEV-DO wireless data communication standard [3], [4].

Packet header compression methods exploit the correlation between successive packet headers, in a manner similar to DPCM. Thus, packet header compression shares the strengths and weaknesses of DPCM: excellent compression is achieved when headers are strongly correlated (as they usually are), but any errors will propagate and contaminate future packets. Although error control exists at the physical layer, some residual packet errors are inevitable, due to channel variations, power control, multi-user scheduling, as well as other issues. Our focus is on the packet losses that are transferred to higher layers and cause error propagation.

The past remedies for the header error propagation fall into two general categories. In uni-directional links, existing

schemes perform periodic refresh with uncompressed headers, thus periodically restoring synchronization between the transmitter and receiver. In bi-directional links, variations of ARQ with partial retransmission of the compressed information have been proposed. An overview of header compression methodologies in uni-directional and bi-directional links is presented in Section II.

The refresh method for the uni-directional link is far from ideal. At high refresh rates, compression ratio is quickly eroded; at lower refresh rates, packet loss ensues. Motivated by the bursty nature of the losses in this application, and by the excellent burst-error correcting capability of Reed-Solomon (RS) codes, we propose to adapt systematic RS codes for the packet header compression problem in the uni-directional links. The distribution of parity bits over the packets is carefully designed to comply with existing protocols and to reduce the overall error rate. Significant improvements over previous solutions are obtained using this method.¹

We also design a reduced-complexity solution for the same channel using an interleaved convolutional code. The reduced complexity is obtained at the cost of a higher residual packet loss rate, compared with the RS code.

We then turn to the bi-directional link, where feedback can be used to acknowledge the receipt of packets. Again we propose to use forward error correction (FEC) methods, where (in principle) multiple packet headers will contribute to each parity bit. Therefore each of the (compressed) packet headers correspond to several symbols in our codewords, and the reverse link provides feedback several times *within* each codeword. This is an unusual situation for code design, and presents a challenge to use the feedback link effectively without cannibalizing the coding gain. We construct a *predictive hybrid ARQ* with multiple retransmission within a convolutional codeword. We also design and use a delay-limited interleaver. Using this structure, very attractive throughput-delay tradeoffs can be obtained in the presence of noise in both forward and reverse links.

We also investigate the interaction of the proposed solution with higher layers, by performing video transfer experiments (simulations) on the coded packet header compression channel.

II. HEADER COMPRESSION METHODOLOGY

Header fields of consecutive packets are not unrelated [5]. For example, consecutive packets in a session share the same source and destination address, so one may economize by sending them only once. The fields whose values are constant

Paper approved by T.-S. P. Yum, the Editor for Packet Access and Switching of the IEEE Communications Society. Manuscript received November 15, 2006; revised September 28, 2007. This work appeared in part in Globecom 2005 and was supported in part by the NSF grant CNS-0627106.

V. Suryavanshi is with Qualcomm Inc., San Diego, CA, USA (e-mail: vsuryava@qualcomm.com).

A. Nosratinia is with the University of Texas at Dallas, Richardson, TX 75080, USA (e-mail: aria@utdallas.edu).

Digital Object Identifier 10.1109/TCOMM.2008.060630

¹Channel coding requires parity bits that slightly erode the compression ratio, but the encoded system still has a strong overall compression ratio while being much more robust.

Fig. 1. The error propagation problem.

throughout a session are called *static* fields. Some other parts of the header are more or less unrelated from one packet to the next, e.g., the Time To Live (TTL), and are known as *random* fields. Finally, there are some parts, such as the packet number, that are not the same, but can be determined from past values. These are known as *inferred* fields.

Existing header compression methods use variations of (nonlinear) DPCM. Random fields are transmitted intact, while static and inferred fields are compressed by reference to the previous packet header(s). The entire process is a reversible (lossless) compression that is characterized by a finite state machine, whose state must be maintained and updated regularly at the compressor and decompressor alike. This state is known as *CONTEXT*.

It is well known that DPCM is a simple and powerful technique that can compress highly correlated sequences. In fact the best header compression techniques can compress a 40-byte IPv4 header down to 2–3 bytes. However the basic DPCM assumes an error-free link; any errors in DPCM transmission will de-synchronize the compressor and decompressor states, thus future received values will be interpreted incorrectly. This problem is known as *error propagation*. (See Figure 1)

The problem of error propagation was recognized in the earliest works on header compression. The method of Van Jacobson [5] proposed periodic refresh to limit the damage caused by channel errors.

Nevertheless, this method concedes lost packets between two refreshes. Perkins and Mutka [6] propose a more robust method where *CONTEXT* is stationary and does not change, i.e., all compression is done with respect to the last uncompressed packet header. Eventually, as differences between the current packet and last uncompressed packet grow, sending the differential may be unprofitable. Thus Calveras *et al* [7], [8] optimized the frequency of transmission of uncompressed headers in the method of [6]. Unfortunately these methods, while removing dependence among compressed headers, lose a significant part of the compression ratio.

Degermark *et al.* [9] observed that the statistics of the compressed headers are not random, but that some residual redundancy remains in them. Often times, the compressed headers are the same from one packet to the next. Thus, whenever a compressed header is lost, for the next packet, the *CONTEXT* is repaired by applying the received delta value twice, thus leading to the TWICE algorithm. The basic assumption is that the lost compressed header was the same as the next one received. This is a similar idea to what is known as *error concealment* in the image and video compression literature. The TWICE update is verified by the checksum of the decompressed header at the transport layer (TCP or UDP).

The basic ideas arising from these methods were adopted by the Internet Engineering Task Force (IETF) into various standard documents (RFC's). Compression of RTP packets via the basic Van Jacobson algorithm with the TWICE decoder is incorporated into RFC 2508 [10]. The Enhanced Compressed RTP (ECRTP) algorithm allows for multiple transmission of compressed information based on channel conditions, and is incorporated into RFC 3545 [11].

Perhaps the most effective header compression standard to date is known as RObust Header Compression (ROHC) RFC 3095 [12]. The fundamentals of ROHC are similar to previous header compression algorithms, but ROHC, through attention to detail, achieves superior compression. Since ROHC is widely considered to be the state of the art, we briefly describe some of its key aspects. Due to space constraints the concepts are presented in a simplified form; the interested reader is invited to consult [12] for a more detailed treatment. The reader who is familiar with ROHC may safely bypass the remainder of this section.

ROHC is in fact not one compression algorithm, but several compression algorithms bundled into one package. There are four *compression profiles* in ROHC (Uncompressed profile, RTP, UDP, and ESP profiles) each of them applies to a particular type of packet stream.

Also, based on the existence of feedback and/or reliability of the channel, ROHC operates in one of several *modes*. In unidirectional links, we have the unidirectional mode or U-mode, also known as ROHC-U. In the presence of feedback there are two modes, the reliable mode (ROHC-R) which is a conservative compression, and optimistic mode (ROHC-O), which has higher compression ratio.

Profiles and modes determine the details of the compression algorithm, so in principle, we can think of ROHC as a family of algorithms. The *profiles* adapt ROHC to the properties of the source, while the *modes* adapt ROHC to the properties of the channel.

ROHC senses the quality of the channel and moves between the different modes via an algorithm described by a finite state machine. Specifically, ROHC starts in the U-mode, and progresses to R-mode and O-mode. Whenever there are too many errors, it falls back down through the mode hierarchy. Furthermore, in each of the modes, the operation of ROHC compressor and decompressor is characterized by a finite state machine. To summarize, there is one small-scale FSM (Finite State Machine) describing the operation of the compression algorithm, while there is (another) large-scale FSM driving the movement between different modes. The interested reader is referred to [12], [13] for a more detailed treatment of these subjects.

III. UNI-DIRECTIONAL LINKS

Several RFCs on header compression operate over half duplex channels. Notably, header compression over simplex channels (RFC 1144) and Uni-directional mode (RoHC). Examples of half duplex communications include IP multi-cast where feedback is not possible and Push-to-Talk over Cellular (PoC). In these cases feedback is not available either due to bandwidth considerations or lack of feedback channel. In IP multi-cast to avoid feedback implosion at the multi-cast server, feedback is disabled completely. One obvious example is real-time streaming where feedback is of no help.

In the absence of feedback, a compressed-header packet must be discarded if any of the preceding packets are lost. With forward error correction (FEC), many of these otherwise lost packets can be recovered. In this section we present FEC techniques based on Reed-Solomon codes and convolutional codes for the header compression in a uni-directional link.

A. Reed-Solomon Codes

There are two aspects of the header compression problem that give rise to bursts of errors. First, each packet header, even when compressed, consists of multiple bytes, thus the loss of each compressed header will result in the loss of 2-5 consecutive bytes. Second, the bursty nature of many channels of interest, e.g., the fading wireless channel, result in consecutive losses. These two factors together lead to a bursty-error (equivalent) channel in our problem.

Motivated by the excellent burst-error correction properties of Reed-Solomon (RS) codes, we use them for the packet header compression problem. A systematic RS encoder accepts K data symbols and generates $N - K$ parity symbols [14]. A symbol is made up of m bits and the maximum codeword length, $N \leq N_{max} = 2^m - 1$. In this paper we consider $m = 8$, i.e., each symbol is one byte and therefore $N_{max} = 255$. At the receiver, a RS decoder can recover these K data symbols if *any* K from N transmitted symbols are received correctly.

In a packet header compression system, periodically an uncompressed header is transmitted that is around 40 bytes long, and following that compressed packets are transmitted that are much shorter. Our encoder will combine the uncompressed and the compressed headers into one group of symbols, and calculates the parity bits for them (see Figure 2). To remain compatible with protocols already in place, we propose to use a systematic code, so that the existing (compressed) packet headers will be transmitted as they are. Additional parity bits will be generated and loaded onto the packets in a manner to be described below.

For a configuration of parity bits that will achieve good error performance, careful attention must be paid to how symbol losses are generated through packet losses, since the distribution of symbols and errors in this application is different from many of the channels where RS codes have been historically used. In particular, note that our symbol errors are highly correlated, even with i.i.d. packet losses, because each packet loss will entail the loss of several systematic bits (compressed header) as well as parity bits that may be loaded onto the packet.

Fig. 2. Generation and distribution of parity symbols across packets. Parities are distributed equally among compressed headers. There are α compressed headers in each batch.

Since RS decoding error is determined by the total number of lost symbols (systematic + parity), the best course of action is to equalize the total number of systematic plus parity bits on each packet.² Therefore, since uncompressed headers are many times larger than compressed headers, parity bits must be loaded onto the compressed-header packets (see Figure 2) in a manner such that the total number of header bytes plus parity bytes for all packets are equal (as close as possible). This is justified because the *marginal* probability of error for all packets, according to the loss model, is identical (which means that we do not have any information that would make the loss of any specific packet distinct from any other). Thus there is no reason for any packet to carry more information (header plus parity) than others.

Assuming an uncompressed header of u bytes and compressed header with c bytes, and α compressed-headers packets in each cycle, the RS encoder accepts $u + \alpha c$ bytes and generates parity symbols as shown in Figure 2. The parity bits are distributed equally among α packets with compressed headers.

I.I.D. Channel: a packet is lost independently of other trans-

²With an underlying assumption that packet loss probability is equal among different type of packets.

Fig. 3. Performance of RS coded system on iid unidirectional channel

mitted packets with probability p . We consider a window of $\alpha + 1$ packets consisting of an 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:

$$P(k) = \begin{cases} (1-p)^{(\alpha+1)} & k = 0 \\ p(1-p)^{(\alpha+1-k)} & k = 1, \dots, \alpha+1 \end{cases} \quad (1)$$

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

$$\bar{m} = \sum_{k=0}^{\alpha+1} k(1-p)^{(\alpha+1-k)}p = (\alpha+1) - \frac{1-p}{p} \left[1 - (1-p)^{\alpha+1} \right] \quad (2)$$

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:

$$\bar{M} = \left\lfloor \frac{t+\alpha}{\alpha+1} \right\rfloor \left((\alpha+1) - \frac{1-p}{p} [1 - (1-p)^{\alpha+1}] \right) \quad (3)$$

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

Correlated Channel: For analysis we use a two-state Gilbert-Elliott model [15], [16] with 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’. We show by P_{ij} the Markov transition probabilities and by P_i the marginal probabilities, i.e., P_1 is the probability of the packet loss. The average number of consecutive packets lost (error burst) is given by $B_L = \frac{1}{P_{10}}$.

The average number of packets discarded in a *window* of $\alpha + 1$ packets can be calculated as:

$$\bar{m} = \sum_{k=0}^{\alpha+1} kP(k) = (\alpha+1) - P_0 [P_{01} [1 - (P_{00})^\alpha] + 1] \quad (4)$$

According to the literature, the i.i.d. packet loss model is reasonable for the wired Internet [17], [18]. In particular, simulations in [18] show that burst length of one (i.e., random

Fig. 4. Performance of RS coded system on a bursty unidirectional channel

packet errors) occur more frequently than burst losses of higher length. Also [17] corroborates this viewpoint 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.

Figure 3 shows the performance of various (N, K) RS codes where $u = 40$ bytes and $c = 2$ bytes for the i.i.d. channel. The proposed coded scheme performs well compared to the uncoded system. Decreasing the code rate provides better protection and hence better performance. Figure 4 shows the performance in a bursty error channel. Other experiments were also undertaken in bursty error channels, which are omitted for the sake of brevity. The interested reader is referred for further experimental results to [19].

B. Convolutional Codes

In the context of our application, the average delay of the Reed-Solomon decoding grows proportionally with the codeword length (see appendix). For delay-sensitive scenarios we may need to seek alternative coding methods. Furthermore, Reed-Solomon codes have decoding complexity that is quadratic in codeword length, so for low-complexity applications, we may look for an alternative.

Convolutional codes can be decoded with delay that is roughly on the order of the constraint length of the code, which is independent (and often much smaller than) the length of the codeword, and have complexity that increases only linearly with codeword length. But due to their limited memory, convolutional codes do not perform well in the presence of successive (bursty) errors, so interleaving must be employed to randomize the errors. Interleaver-imposed delay is an issue that will be addressed in the sequel.

As shown in Figure 5 at the transmitter a total of $u + \alpha c$ bits are fed to the interleaver. The output bits from the interleaver are fed to a Recursive Systematic Convolutional (RSC) encoder. Parity bits coming out of this encoder are fed to the interleaver and the output interleaved parity bits are equally divided among α compressed packet headers. Interleaving data bits ($u + \alpha c$ of them) before encoding ensures that the parity bits generated belong to data bits which are at least some samples apart. Correct reception of one such

Fig. 5. The header compression system with convolutional codes

Fig. 6. Performance of interleaved and convolutionally coded system over iid unidirectional channel

parity bit can contribute in reconstruction of many packets. The parity bits are also interleaved so that when a packet is lost, the missing parity bits are separated as much as possible in the codeword. Also the parity bits are equally distributed among the α compressed packets instead of transporting them in one single packet. This excludes the possibility of losing all the parity bits except when all the packets are lost (a rare event). Similarly decoding is done as shown in Figure 5 at the receiver side.

In our simulations $u = 40$ bytes and $c = 2$ bytes. We compare our scheme with ECRTTP and ROHC operating in uni-directional mode over an i.i.d. channel. The RSC encoder given in [20] with rate $1/2$, $K = 6$ and generator matrix $g^0 = [101011]$ and $g^1 = [111101]$ is used. The results are shown in Figure 6. Extensive experiments were also undertaken in the bursty error channels, the results of which cannot appear here due to editorial constraints on the number of figures. The interested reader is referred for further experimental results to [21].

C. Delay-Limited Interleaver Design

An interleaver π is characterized by a pair of spreading factors, indicating the distance properties of symbols at the

input and output. The symmetric expression of spreading factor is the s -parameter, which is the largest number s such that symbols within distance s at the input are s samples apart in the output stream [22]. The interleaver delay is defined:

$$\delta = \max_i (\pi(i) - i) \quad (5)$$

Let $\mathcal{D}_{max}$ be the maximum allowable delay of the system (in bits). From (5) it can be seen that if an interleaver needs to be designed with $\mathcal{D}_{max}$ in view, then

$$\pi(i) \leq \mathcal{D}_{max} + i \quad (6)$$

The delay-limited interleaver design algorithm can be described as follows. First, an initial guess for the s -parameter is made. A necessary and sufficient condition for an interleaver with spread s and period P to exist is that $P \geq s^2$ [23] therefore our initial guess for s is $\lfloor \sqrt{P} \rfloor$. The algorithm tries to design the interleaver with the specified delay constraint and s parameter. If at any step of the algorithm the interleaver cannot satisfy the delay constraint for a given s , the s parameter is decreased by 1 and the algorithm starts all over again. The interleaver is designed similarly to the classic block interleaver but adhering to Equation (6).

As in the case of a block interleaver, we start by setting $J_{11} = 1$, i.e., the element on the upper left corner of the matrix is one. Recall that a column represents the input and a row represents the output of the interleaver, so every time a 1 is placed, the other elements in the corresponding row and column are set to zero. The next 1 is placed in the second row and $1 + (s+1)^{th}$ column. Similarly for the third output bit of the interleaver, a 1 is placed in the third row and $2 + (s+1)^{th}$ column. At some point, we will need to wrap back once we exceed the delay limit or when the column value exceeds the period P of the interleaver. In this case we place a 1 in the next least valued and unoccupied column. We continue to fill the matrix in a similar fashion.

The resulting interleaver matrix will have a block interleaver structure but some of the permutation indices are not allowed. The non-permissible indices are represented by x. Thus the interleaver matrix is filled as in the case of a block interleaver but will have delay less than $\mathcal{D}_{max}$. Below we show a toy example of the interleaver matrix $\mathbf{J}_{con}$ with spread $s = 1$ and

Fig. 7. Predictive hybrid ARQ – the blocks entitled “encoder” and “decoder” are delineated in Figure 5

maximum delay $\mathcal{D}_{max} = 7$ bits. The resulting $\mathbf{J}_{con}$ will have a trapezoidal appearance due to the delay constraint.

[illegible]

IV. BI-DIRECTIONAL LINKS

In this section we present a coded ARQ technique for resilient packet header compression in the bi-directional links. Because the network ARQ function is available on a per-packet basis, while the codewords consist of multiple packets, it follows that in our system, multiple opportunities exist within a codeword to acknowledge partial receipt of codeword contents. Based on this partial information, a decision must be made on transmission of additional data and parity. Therefore, a new system of ARQ is needed to address this particular situation, which we refer to as *predictive hybrid ARQ*.

Classic ARQ achieves reliability by retransmissions of lost data [14], [24]. The ability to retransmit means that there are no packet losses in a delay-unlimited system, however, when the channel deteriorates, multiple retransmissions reduce the throughput of ARQ. To reduce the number of retransmissions the forward link can be encoded, a method known as hybrid ARQ. If the error pattern is not correctable, the receiver can then request retransmission of the erroneous block, thus maintaining the reliability of the system [25].

In the existing hybrid ARQ schemes [14] the decoder waits until a codeword is fully received, then tries to decode the codeword and see if retransmission is necessary. In our application, the codeword consists of several packet headers. We built our systems on top of existing link-layer and possibly transport-layer ARQ's, therefore ACK/NACK will be transmitted for each packet, which is only part of the codeword for us. Thus, we have a system where acknowledgments are available for parts of the codeword, not for the entire codeword. This is a new scenario and requires a fresh approach that can take advantage of the partial information about the state of the decoder.

Our literature search did not find any method that would match the requirements of our application, but we note [26] as an interesting case study. In this reference, the transmitted packet includes additional information (partial checksums) so that the rough location of error(s) can be determined at the decoder. Then, the acknowledgment will ask only for the corrupted data to be retransmitted. In this scheme the transmitter and receiver exchange more information than the traditional ARQ, however, still only one ACK/NACK is available per encoded block, therefore, as mentioned earlier, it does not match our application.

In our scheme, feedback provides the encoder with some information about the status of the decoder *while the codeword is being transmitted*. Each NACK means that one packet (and thus part of our codeword) was lost. Then, the encoder makes an estimate of whether the decoder can continue to decode despite the error, i.e., whether the decoder can recover from this error using parity bits. If the answer is yes, then coded transmission will continue. Otherwise, the last missing packet will be retransmitted.

The block diagram of the system is shown in Figure 7. The encoder and decoder block are shown in Figure 5. Each time a packet is lost, the decoder transmits a NACK to the encoder. Once a NACK is received, the transmitter will mimic the decoding operation and will determine if this packet loss will result in a decoding failure at the receiver. If so, it will retransmit the packet. Due to non-zero round-trip time, a buffer is provided at the encoder to allow for potential retransmissions.

For our simulations, the packet headers have $u = 40$ bytes and compressed headers have $c = 2$ bytes. We use the RSC encoder given in [20] with rate $1/2$, $K = 3$ and generator matrix $g^0 = [101]$ and $g^1 = [111]$. In all experiments the decoder timeouts after a period equivalent to round trip-time of a packet and retransmits the NACK. The feedback channel is assumed i.i.d. with a packet loss rate of 0.1. We use delay-constrained interleavers with $D_{max} = 200, 400, 600$ and 900 bits, designed with the algorithm of Section III-C.

Figure 8 demonstrates the performance of the proposed predictive hybrid ARQ technique over a correlated (bursty error) channel. As expected, the average delay increases with D_{max} but is lower than ROHC-O mode for most channel conditions. The spreading factor s of the interleaver increases as D_{max} increases, allowing for better error correction. This is evident from Figure 8 where the throughput improves as

Fig. 8. Delay and throughput, correlated bidirectional channel with burst length, $B_L = 5$

D_{max} increases. Experiments were also performed on the i.i.d. channel, whose results can be found in [27]. The advantages are especially evident for throughput in the low to medium-loss scenarios, and for delay in medium to high packet-loss scenarios.

To determine the effect of resilient packet header compression on the application layer, we looked at packet video transfer. We transmitted a segment of the *Foreman* QCIF color video sequence encoded at 10 fps resulting in a 64 kbps compressed bit stream through a lossy channel with packet compression. At the other end, after packet decompression, the video is decompressed and the Peak Signal-to-Noise Ratio (PSNR) calculated. Here we compare the PSNR performance of our scheme with a no header compression, and header compression with unidirectional 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 9 presents the PSNR performance for various packet loss probabilities and an average burst length of 5 packets. At high packet loss rates, uncoded (conventional) header compression performs worse than no header compression case. We observe an average PSNR gain of 2 dB to 3.5 dB comparing to no header compression, or insufficiently protected header compression.

For Figures 3 and 4, the 95% confidence intervals are within $\pm 0.1\%$ of each reported value, for Figure 6 within $\pm 1.44\%$ of each reported value, and for Figure 8 within $\pm 1.30\%$ for throughput, and within $\pm 0.60\%$ for delay values.

V. CONCLUSION

This paper presents error-resilient methods for packet header compression for uni-directional and bi-directional links. Reed-Solomon encoding as well as convolutional coding solutions are studied for the uni-directional link. In bi-directional links, a system involving convolutional codes combined with a new adaptive hybrid ARQ system is proposed.

APPENDIX A DELAY CALCULATIONS

We calculate the average delay when using a Reed-Solomon codes under i.i.d. channel errors. Assume a (N, K) RS code,

Fig. 9. Comparison on *Foreman* video sequence

take n_f to be the position of first channel error, and Δ to be the number of symbols in error prior to successful decoding. One must wait for decoding until exactly K correct symbols have been received, and the location at which this happens is denoted by S . Note that symbols before n_f have zero decoding delay and symbols after S have no effect on decoding. Packets between n_f and S have delay $S - n$. The average delay can be calculated as:

$$\bar{D} = \frac{1}{N} \sum_{n=n_f}^S (S - n) \quad (7)$$

After substituting $S = K + \Delta$ the expected overall delay per packet $E[\bar{D}]$ is:

$$\begin{aligned} E[\bar{D}] &= E \left[\frac{(K + \Delta - n_f + 1)(K + \Delta - n_f)}{2N} \right] \\ &\geq E \left[\frac{K(K + \Delta - n_f)}{2N} \right] \end{aligned} \quad (8)$$

where p is the probability of packet loss and r is the code rate. The position of first error n_f is a geometric random variable with mean $1/p$. Also, to successfully decode a codeword, we need K correct symbols, thus the “time to K successes,”

which is equivalent to $K + \Delta$, follows a negative binomial distribution, therefore

$$P[K + \Delta = m] = \binom{n-1}{K-1} (1-p)^K p^{m-K} \quad m = K, K+1, \dots$$

or

$$P[\Delta = \delta] = \binom{n-1}{K-1} (1-p)^K p^\delta \quad \delta = 0, 1, \dots$$

The mean of Δ can be calculated to be $E[\Delta] = \frac{K}{1-p} - K$. Therefore the bound on the delay is:

$$E[\bar{D}] \geq \frac{r}{2} \left(\frac{Nr}{1-p} - \frac{1}{p} \right) \quad (9)$$

REFERENCES

- [1] CAIDA, "Packet length distribution," Aug. 2004, http://www.caida.org/analysis/AIX/plen_hist/.
- [2] Sprint Applied Research Group, "Packet trace analysis," Jan. 2005, <http://ipmon.sprint.com/packstat/packetoverview.php>.
- [3] M. Yavuz, S. Diaz, R. Kapoor, M. Grob, P. Black, Y. Tokgoz, and C. Lott, "VoIP over CDMA2000 1xEV-DO revision A," *IEEE Commun. Mag.*, vol. 44, no. 2, pp. 50–57, Feb. 2006.
- [4] Q. Bi, P. Chen, Y. Yang, and Q. Zhang, "An analysis of VoIP services using 1xEV-DO revision A system," *IEEE J. Select. Areas Commun.*, vol. 24, no. 1, pp. 36–45, Jan. 2006.
- [5] 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>.
- [6] S. J. Perkins and M. W. Mutka, "Dependency removal for transport protocol header compression over noisy channels," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 2, June 1997, pp. 1025–1029.
- [7] A. Calveras, M. Arnau, and J. Paradells, "A controlled overhead for TCP/IP header compression algorithm over wireless links," in *Proc. 11th International Conference on Wireless Communications*, Calgary, Canada, 1999.
- [8] —, "An improvement of TCP/IP header compression algorithm for wireless links," in *Proc. 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, Orlando, FL, July 1999, pp. 39–46.
- [9] M. Degermak, M. Engan, B. Nordgren, and S. Pink, "Low loss TCP/IP header compression for wireless networks," in *Proc. ACM Mobicom 96*, New York, NY, Oct. 1997.
- [10] 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>.
- [11] T. Koren and *et al.*, "RFC 3545 - enhanced compressed RTP ECRTT for links with high delay, packet loss and reordering," RFC 3545 IETF Network Working Group, July 2003, <http://www.ietf.org/rfc/rfc3545.txt>.
- [12] C. Bormann and *et al.*, "RFC 3095-RObust Header Compression ROHC: Framework and four profiles: RTP,UDP,ESP, and uncompressed," RFC 3095 IETF Network Working Group, July 2001, <http://www.ietf.org/rfc/rfc3095.txt>.
- [13] E. Ertekin and C. Christou, "Internet protocol header compression, robust header compression, and their applicability in the global information grid," *IEEE Commun. Mag.*, vol. 42, pp. 106–116, Nov. 2004.
- [14] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.
- [15] E. N. Gilbert, "Capacity of a burst noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sept. 1960.
- [16] E. O. Elliot, "A model of the switched telephone network for data communications," *Bell Syst. Tech. J.*, vol. 44, no. 1, pp. 89–109, Jan. 1965.
- [17] M. S. Borella, S. Uludag, G. Brewster, and D. Swider, "Internet packet loss: measurement and implications for end-to-end QoS," in *Proc. International Conference on Parallel Processing Workshops*, Jan. 1998, pp. 3–12.
- [18] M. Arai, A. Chiba, and K. Iwasaki, "Measurement and modeling of burst packet losses in internet end-to-end communications," in *Proc. Pacific Rim International Symposium on Dependable Computing*, Dec. 1999, pp. 260–267.
- [19] V. Suryavanshi, A. Nosratinia, and R. Vedantham, "Resilient packet header compression through coding," in *Proc. IEEE GLOBECOM*, vol. 3, Dallas, TX USA, 29 Nov.-3 Dec. 2004, pp. 1635–1639.
- [20] P. Frenger, P. Orten, and T. Ottosson, "Convolutional codes with optimum distance spectrum," *IEEE Commun. Lett.*, vol. 3, pp. 317–319, Nov. 1999.
- [21] V. Suryavanshi and A. Nosratinia, "Convolutional coding for resilient packet header compression," in *Proc. IEEE GLOBECOM*, Nov. 2005, pp. 771–775.
- [22] C. Heegard and S. B. Wicker, *Turbo Coding*. Kluwer International Series in Engineering and Computer Science, 1999.
- [23] V. Tarokh and B. Hochwald, "Existence and construction of block interleavers," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 25, no. 1, Apr. 2002, pp. 1855–1857.
- [24] S. Lin, D. J. Costello, and M. Miller, "Automatic repeat request error control schemes," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 15–17, Dec. 1984.
- [25] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Pearson-Prentice-Hall, 2004.
- [26] H. S. Cheng, G. Fairhurst, and N. Samaraweera, "Efficient partial retransmission arq strategy with error detection codes by feedback channel," in *IEE Proc. - Commun.*, vol. 147, no. 5, Oct. 2000, pp. 263–268.
- [27] V. Suryavanshi and A. Nosratinia, "A hybrid ARQ scheme for resilient packet header compression," in *Proc. Asilomar Conference on Signals, Systems and Computers*, Oct. 2005, pp. 36–40.

Vijay Suryavanshi received his B.E. from the University of Mumbai in 2001 and M.S in Electrical Engineering from the University of Texas at Dallas in 2005. He interned at Nokia Research Center, Irving, Texas for a year until May 2005 and joined Qualcomm Inc. San Diego in April 2006 as a Systems Engineer in QChat. His main interests are in the field of real-time media delivery over wireless IP networks particularly designing protocols for media applications.

Aria Nosratinia is professor of Electrical Engineering at the University of Texas at Dallas. He received his PhD in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 1996. He has held visiting positions at Princeton University, Rice University, and UCLA. His research interests are in the general area of signal processing, coding and information theory, with applications to wireless communications. He is associate editor for IEEE TRANSACTIONS ON INFORMATION THEORY, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE SIGNAL PROCESSING LETTERS, and IEEE WIRELESS COMMUNICATIONS. He has also served as associate editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING (2002-2008). He is a recipient of the National Science Foundation career award, as well as two chapter awards for service to the IEEE Signal Processing Society.