Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2009, Article ID 681078, 10 pages
doi:10.1155/2009/681078

Research Article

Adaptive Error Resilience for Video Streaming

Lakshmi R. Siruvuri, Paul Salama, and Dongsoo S. Kim

Department of Electrical and Computer Engineering, Purdue School of Engineering and Technology, Indiana University-Purdue University at Indianapolis, 723 West Michigan Street, SL160, Indianapolis, IN 46202, USA

Correspondence should be addressed to Paul Salama, psalama@iupui.edu

Received 1 July 2008; Revised 29 January 2009; Accepted 24 March 2009

Recommended by Lorenzo Ciccarelli

Compressed video sequences are vulnerable to channel errors, to the extent that minor errors and/or small losses can result in substantial degradation. Thus, protecting compressed data against channel errors is imperative. The use of channel coding schemes can be effective in reducing the impact of channel errors, although this requires that extra parity bits to be transmitted, thus utilizing more bandwidth. However, this can be ameliorated if the transmitter can tailor the parity data rate based on its knowledge regarding current channel conditions. This can be achieved via feedback from the receiver to the transmitter. This paper describes a channel emulation system comprised of a server/proxy/client combination that utilizes feedback from the client to adapt the number of Reed-Solomon parity symbols used to protect compressed video sequences against channel errors.

Copyright © 2009 Lakshmi R. Siruvuri et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

With the advances in technology, applications such as video telephony, video conferencing, video-on-demand, video broadcasting, and video email have become a reality. In fact, during the last thirty years, there has been a tremendous advance in the visual communication field [1]. One of the drawbacks of digital video signals is that they occupy too much bandwidth, and thus compressing video prior to storage and/or transmission is necessary. This has led to the development of many successful video compression standards [1–9].

Compressed video is very vulnerable to channel errors, and different means have to be used to protect the compressed data [10]. These can be classified into three categories: error resilience, error control coding, and error concealment. Error resilience refers to schemes that introduce error resilient elements at the video compression stage, which reduce the interdependencies of the data-stream, in order to mitigate error propagation. Inevitably, the use of error resilience degrades coding efficiency.

Error control coding refers to error protection by using channel coding schemes in the application layer. Contrary to source coding, channel coding inserts redundant information into the data-stream, which helps detect and correct errors due to the channel impairments. When video signals are transmitted over packet networks or wireless channels, serious corruption might occur to the data-stream due to (1) burst packet losses that stem from network congestion and/or (2) burst bit errors that are a consequence of channel fading. Hence, it is wise to introduce channel coding in the application layer to obtain better error protection. One particular channel coding scheme, namely, Reed-Solomon (RS) coding, is effective since RS codes are maximum distance separable codes that are well suited for error protection against burst errors.

It is possible to jointly use error resilience and error control schemes, commonly referred to as joint source and channel coding. This requires two types of optimized data rate allocations: (1) the optimized data rate allocation between source coding and channel coding, and (2) the optimized allocation of the source coding data rate among the various source coding elements to introduce an appropriate amount of error resilience into the data-stream. Unlike the above schemes that actively place error protection at the encoder side, error concealment is a passive scheme that

attempts to perform error recovery at the decoder by using actual video content, where that content is correctly decoded pixel values and motion vectors.

In general, the performance of source coding or joint source and channel coding schemes can be further improved through the use of feedback. In [11] a Rate-Distortion (R-D) framework for rate-constrained motion estimation and mode decision based on the use of several reference frames is proposed. In addition, a feedback channel is used by the decoder to inform the encoder about successful or unsuccessful transmission events by sending positive (ACK) or negative (NACK) acknowledgments. This information is utilized for updating the error estimates at the encoder, which the encoder utilizes in the compression of subsequent frames. The disadvantages of this method are: (1) it is not standard compliant, (2) feedback is assumed to be received in a timely manner, and (3) the increase in complexity resulting from having to search for optimal modes and motion vectors over multiple reference frames. In addition, it assumed that two subsequent error events are unlikely to occur.

Alternatively, [12] presents feedback and error-protection strategies for wireless transmission of progressively coded video using an unequal error protection scheme that employs high-memory rate-compatible punctured convolutional codes with a sequential decoding algorithm. This method assumes that the transmitter knows perfectly the long term channel statistics as well as the existence of an error-free low bit rate channel in both directions. The feedback channel is used for sending feedback via several ARQ strategies, but the paper does not describe what happens when feedback is delayed long enough to be useful.

In both [11, 12], the R-D framework does not take into consideration distortion due to data corruption and/or loss. Recent methods that take into consideration end-to-end distortion have been proposed in [13, 14]. In [13], a stochastic framework for an R-D-based encoder design for video compression and transmission over error-prone networks was described. The distortion measure used was the mean square error evaluated over an ensemble of channels of known packet loss probabilities. Furthermore, the channels are assumed to have uniform loss probabilities, and the packet loss probabilities were used in the stochastic R-D optimization framework for selecting the optimal macroblock mode and motion vectors. Similarly, [14] describes an end-to-end method for R-D-optimized selection of the coding modes for H.264 video coding. The proposed scheme assumes that the channel error rates as well as the error concealment method used by the decoder are both known at the encoder. A new Lagrange multiplier was derived, but the derivation of the Lagrange multiplier assumes high-resolution quantization, which does not necessarily apply well for video compressed at low or very low data rates.

The schemes proposed in [11–14] all tackle the problem of optimally coding video (in the R-D sense) for the purpose of transmission, based on feedback from the receiver to the transmitter, but do not address the problem of reliably transmitting previously coded video streams over data channels while utilizing feedback from the receiver to the transmitter. Furthermore, they assume that the channel conditions are perfectly known or that feedback arrives in a timely fashion, which are not always possible. Streaming preencoded video sequences over both constant and variable bit rate channels, such that each video unit is decoded before a deadline was addressed in [15]; however it was assumed that (1) the channel was error free although it had limited transmission rate, (2) the packet losses and delays in the access network can be neglected, and (3) immediate retransmission is available.

In this paper we address the problem of transmitting previously coded H.264 [9] video streams when channel loss probabilities are not exactly known by the transmitter, and when feedback from the receiver to the transmitter does not always arrive in a timely fashion. We first describe a system that emulates a real-time network, and then we propose an adaptive Reed-Solomon error control coding scheme that utilizes feedback from a client to adjust the protection level at the server. The server uses a three-state channel model and the feedback data from the client to estimate the future channel state and consequently the protection level needed.

2. Emulation System

Any communication system consists of a transmitter, a receiver, and a channel through which data is transmitted. In general any communication channel, especially wireless networks, suffers from data loss and/or bit corruption, which affect the quality of data being transmitted. In particular when compressed video sequences are being sent, the various channel impairments will lead to visual distortion when the signal is reconstructed at the receiver. Furthermore, due to the predictive nature of current video compression standards, this distortion will propagate through the video sequence. In fact, based on many factors such as source coding, the transport protocol, and the amount and type of information loss, the distortion induced can range from degradation that may encompass a few frames to a completely unusable video sequence. Therefore it is necessary for the transmitter to attempt to protect the data and for the receiver to perform error concealment at the receiver to minimize any observed distortion.

In order for the transmitter to adequately protect compressed video sequences, it needs to use some form of channel coding. However, in order to avoid wasting bandwidth unnecessarily, the transmitter needs to tailor the protection to the current channel conditions. This can be achieved via feedback messages sent by the receiver that indicate the latest channel loss rates. The advantage of this approach is that the transmitter can attempt to track the channel changes and accordingly adapt the protection it affords to the compressed data stream instead of assuming certain loss probabilities.

In our work we emulate the communication channel by a proxy server that intentionally generates packet delays and losses. In fact, all the transmissions between the server and the client are done via the proxy. Since the receiver needs to convey loss information back to the transmitter

as it receives the incoming video, two independent channels are established. One channel is utilized to reliably transmit control information from the client/receiver to the server/transmitter, while the other is used for the actual transfer of compressed video. The UDP/IP protocol suite is used for the transmission of data packets and the TCP/IP protocol suite for the transmission of control/feedback packets. All the control signals are sent via the TCP ports, and the data signals are sent through the UDP ports. Request signals and feedback signals are considered as control signals.

The process is as follows. The client requests a particular video stream at a particular frame and data rate from the server via the proxy. The proxy, which is waiting for a signal from the client, receives the client's requests, parses the data to be forwarded to the server, makes a connection with the server, and forwards the request. The server receives the request and transmits the required data. The proxy then receives the data sent from the server and forwards it to the client. In the process, the proxy delays both control and data packets and induces loss to the data carrying packets only. The entire process is illustrated in Figure 1.

The server channel encodes the requested stream by applying the appropriate RS coding level and interleaves the channel-coded data. The size of each interleaving block is $N \times S$, where N is length of each RS code word, and the span size S is the number of RS code words that are to be interleaved together. The interleaved blocks are numbered and passed as header information to be used at the client for error detection. The data of size $(K \times S)$, where K is the number of message symbols in each code word, is passed through the RS encoder. The RS encoder adds the necessary parity symbols based on the values of N and K. The number of parity symbols added is $N - K$. In our work, the value of N is always fixed at 255 as each symbol is equal to 8 bits. The value of K is varied according to the packet loss rate estimates obtained based on the feedback from the client.

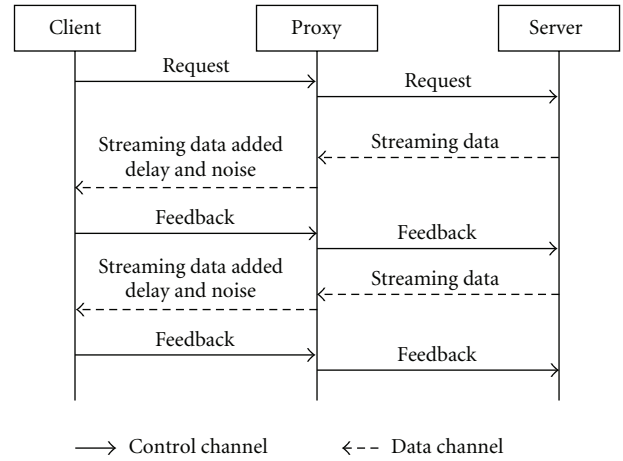


Figure 1: Sequence diagram of the server-proxy-client model.

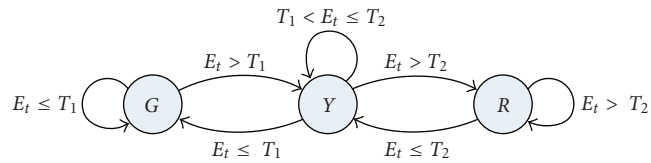


Figure 2: State transition diagram of the adaptive channel coding.

3. Adaptive Reed-Solomon Channel Coding

In order to efficiently utilize the channel and to adapt the transmitted data to varying channel conditions, the server applies varying protection levels, based on feedback from the client, which contains information regarding the number of packets lost in each interleaved block. The server tries to predict the number of losses in the future based on the feedback information received using a weighted moving average and deviation as given in (1):

$$\begin{aligned} \tau_{t+1} &= \alpha x_t + (1-\alpha)\tau_t, \\ \delta_{t+1} &= \beta |x_t - \tau_{t+1}| + (1-\beta)\delta_t, \\ E_t &= \tau_t + c\delta_t. \end{aligned} \tag{1}$$

Here α and β are smoothing factors whose values are 0.4, c is tolerance factor whose value is either 1 or 2, x_t is the number of lost packets sent by the client, E_t the predicted future losses, τ_t the average loss, and δ_t the standard deviation of the number of lost packets. Based on the predicted value E_t, the server changes the message length K, which amounts to changing the error protection.

The process through which the server modifies the protection it applies to data packets is described via a three-state state diagram as shown in Figure 2. Again E_t is the predicted loss at the server based on the feedback information received, and T_1 and T_2 are two preselected thresholds known as the lower threshold level and the upper threshold level, respectively.

When E_t is less than T_1, the server views the current error level as being under control and the system is considered to be in the good state G. At this stage, the message length K is maintained at N-TP/S, where P is the packet size, and T a variable threshold that is set equal to T_1. When the predicted error goes beyond T_1, while the system is in state G, the system then enters state Y, the threshold T incremented step by step towards T_2, and the message length adjusted accordingly. The system remains in state Y as long as the predicted error level is less than or equal to T_2. When E_t exceeds T_2, the system enters state R, where the error level is considered too high. At this point the message length is set to and kept at N-T_2P/S. When the predicted error falls below T_2 while the system is in state R, the threshold T is decremented step by step toward T_1 and the message length is set to N-TP/S accordingly.

4. Results

To test the efficacy of the proposed approach, simulations were run for different video sequences by setting different proxy parameters. The main parameters of interest at the

server are the coded bit rate of the video sequence, packet size, and interleaving span size. The different sequences used were *foreman*, *Akiyo*, *Carphone*, and *Mother-Daughter* in QCIF format. All of the above mentioned sequences have 300 frames except for the *Mother-Daughter* sequence which has 950 frames. A long sequence was made by combining all the sequences, *Akiyo*, *Carphone*, *Foreman*, *Salesman*, *Container*, *Mother-Daughter*, *Coastguard*, and *Claire*. This long sequence named "*all*" has 3315 frames with duration of 110.5 seconds at 30 frames/sec. The sequences were coded at bit rates of 32 kbps, 64 kbps, and 128 kbps. The packet sizes used were 128 bits, 256 bits, 512 bits, and 720 bits. The span sizes used were 8, 16, and 20. Furthermore, each QCIF frame was divided into 9 slices, and an *IPPP*... GOP structure was used, whereby an I frame is inserted every 15 frames.

At the receiver, the decoder attempts to decompress the video sequence after the Reed-Solomon decoder has fixed any channel errors. If the Reed-Solomon decoder cannot fix some errors, the decoder will then partially decode the video stream and will discard whatever it cannot decode until it finds a valid start code. This process is repeated until the entire video stream has been decompressed. As a consequence, some frames within the video sequence will incur damage. The damaged regions in any frame are replaced by the colocated regions in its previous neighbor.

The parameters of interest at the proxy are the delay used and the buffer size at the proxy. The different delays used were constant delay in the range of 0.22–0.35, constant interleaving delay in the range of 0.026–0.03, and random delay with mean in the range of 0.01–0.035. Different buffer sizes used were in the range of 8–1000. In the current work buffer size refers to the size of the priority queue used at the proxy. The experiments were conducted with a given set of parameters using feedback, and the same experiment was conducted without feedback. The values of T_1 and T_2 used in the experiments with feedback are approximately 5% and 20% of the number of packets in an interleaving block assuming that the packet losses in the network range are from 0 to 20%. The maximum number of packets the adaptive algorithm can recover is T_2. The experiments without feedback have message length K set at T_1 (which is usually 5% of the number of packets in an interleaving block), for the entire experiment. All comparisons are based on PSNR values and the total symbol errors that occurred and were corrected in each case. The PSNR values were computed as PSNR = $10\log(255^2/\text{MSE})$, where MSE denotes the mean square error between the decoded uncorrupted frames and those that have been reconstructed using both the adaptive and nonadaptive schemes. A PSNR value of 100 dB or more was used to indicate perfect reconstruction. In other words a reconstructed frame matches exactly with its corresponding decoded uncorrupted counterpart.

Table 1 provides a summary of the different rates, delays, buffer sizes, packet sizes, and span sizes used throughout. Figures 3, 4, and 5 illustrate the performance of the overall system when *foreman* was coded at a data rate of 64 kbps, and the data packets experienced random delay with a mean of 0.057 seconds. The receiver had a buffer size of 40 packets where each packet size was 720 bits. An interleaving span size

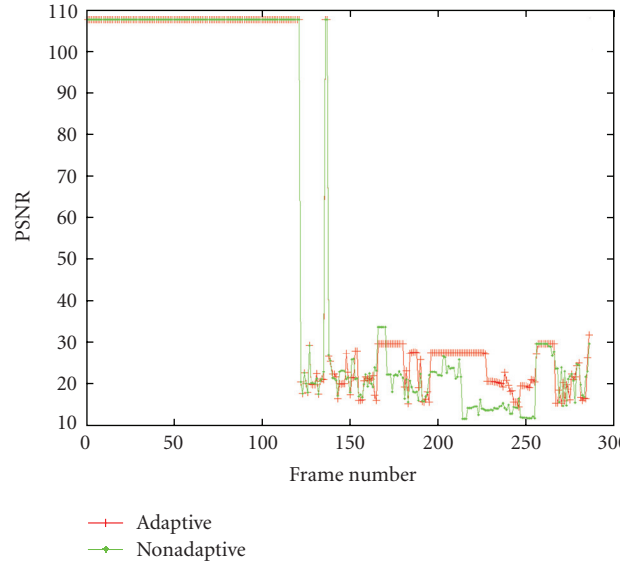


Figure 3: PSNR for "*foreman*" sequence at 64 kbps, random delay 0.057, buffer size 40, packet size 720, span 20, PL rate 30.9%.

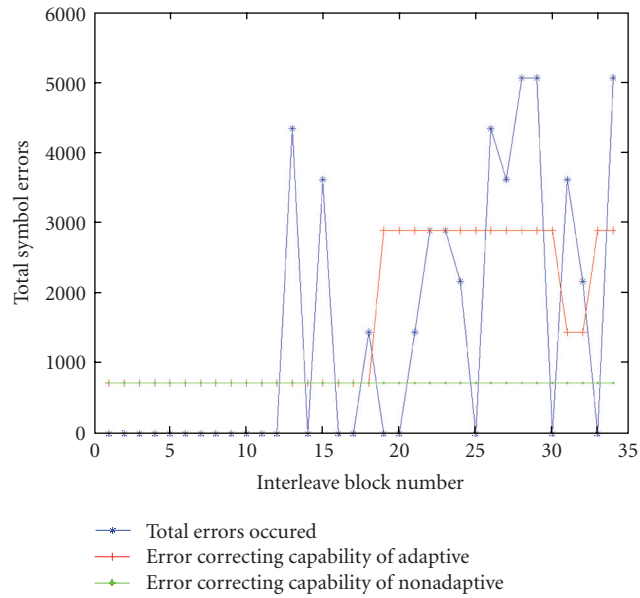


Figure 4: Performance of feedback for "*foreman*" sequence at 64 kbps, random delay 0.057, buffer Size 40, packet size 720, span 20, PL rate 30.9%.

of 20 was used for a packet loss (PL) rate of 30.9%, and the feedback packets were subjected to a constant delay of 0.057 seconds. The PSNR values were better in the adaptive case when compared to the nonadaptive case by about 40.39%, as the adaptive algorithm had the capacity to correct the losses 28.57% of the time. The values of T_1 and T_2 used in this experiment are 1 and 4 which are approximately 5% and 20% of the number of packets in the interleaving block. The nonadaptive algorithm had the capacity to correct the losses 0% of the time. The mean error in predicting the number of packets and hence symbols lost is 1.3529 with a variance

Table 1: Summary of the data rates, delays, buffer sizes, packet sizes, and span sizes for the experiments.

Seq. name-rate-buffer (fig #)	Data rate	Delay	Buffer size	Packet size	Span size
foreman-mid-small (3–5)	64 kbps	0.057 s	40 packets	720 bits	20
all-low-large (6–8)	32 kbps	0.026 s	500 packets	256 bits	16
all-low-small (9–11)	32 kbps	0.028 s	175 packets	256 bits	8
all-mid-mid (12–14)	64 kbps	0.020 s	250 packets	256 bits	16
all-high-large (15–17)	128 kbps	0.012 s	500 packets	256 bits	8

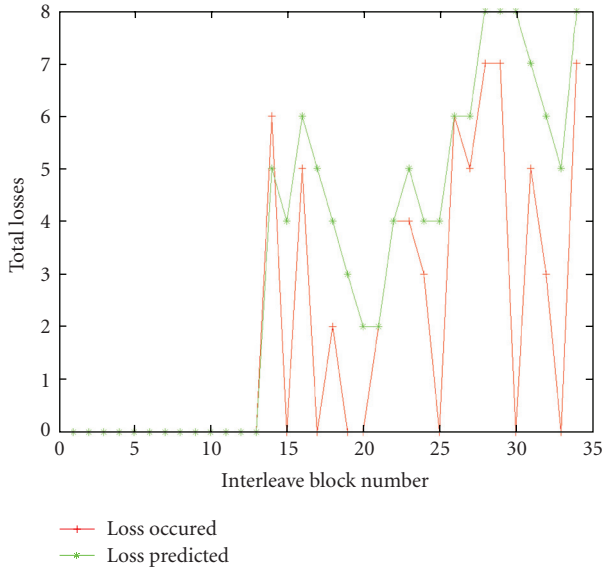


Figure 5: Performance of prediction for "*foreman*" sequence at 64 kbps, random delay 0.057, buffer size 40, packet size 720, span 20, PL rate 30.9%.

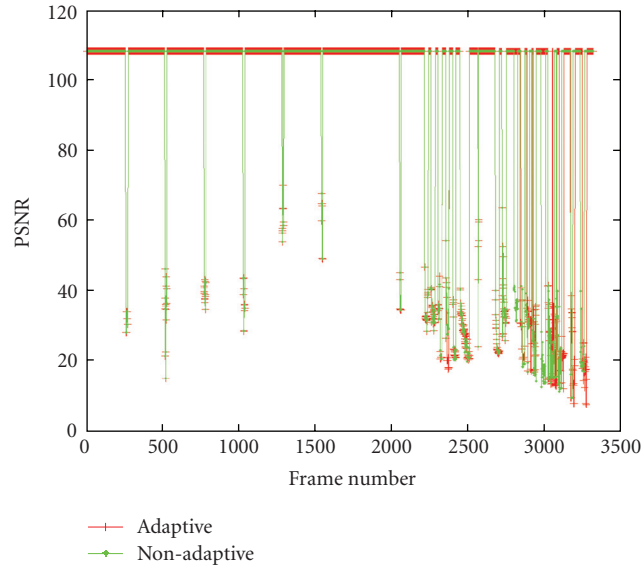


Figure 6: PSNR comparisons for "*all*" sequence at 32 kbps, random delay 0.026, buffer size 500, packet size 256, span 16, PL rate 4.9%.

of 3.6898. Since the maximum number of packets that can be recovered using the current work is T_2, it is not possible to recover losses greater than T_2 even if the predicted loss is always greater than the actual loss. The value of c (prediction parameter) used in this experiment is 2.

Figures 6, 7, and 8 illustrate the performance of the overall system when the "*all*" sequence was coded at 32 kbps, the data packets experienced random delay with a mean of 0.026 seconds, and the packet loss rate was 4.9%. The experiment also used a random delay with a mean of 0.026 seconds for the feedback packets, c was set to 1, and the values of T_1 and T_2 used were 1 and 4 which are approximately 5% and 20% of the number of packets in the interleaving block. The PSNR performance was better in the adaptive case when compared to the nonadaptive case by about 12.54%. The adaptive algorithm had the capacity to correct the losses 17.39% of the time, whereas the nonadaptive algorithm had the capacity to correct the losses 8.7% of the time. The average error in predicting the number of packets and hence symbols lost is 0.99 with a variance of 3.35. From Figure 8, it can be seen that at low packet losses, the adaptive case underestimated the losses most of the time. As a consequence, its error correcting capability was weak. In this case it may have been better to use $c = 2$.

Figures 9, 10, and 11 depict the performance of the overall system when "*all*" sequence was coded 32 kbps, the data packets experienced random delay with a mean of 0.028 seconds, and the packet loss rate was 10.07%. The experiment also used a random delay with a mean of 0.028 seconds for the feedback packets, c was set to 1, and the values of T_1 and T_2 used were 1 and 3 which are approximately 5% and 20% of the number of packets in the interleaving block. The average gain, in PSNR, afforded by the adaptive scheme when compared to the nonadaptive case was about 62.49%. The adaptive algorithm had the capacity to correct the losses 16.88% of the time, and the nonadaptive algorithm had the capacity to correct the losses 7.79% of the time. From Figure 11, it can be seen that the estimation was better than in the previous experiment. It can be deduced that at higher packet losses the prediction is more efficient than at lower packet losses, which is reflected by the fact that the mean error in predicting the number of packets and hence symbols lost is 0.91 with a variance of 1.83.

Figures 12, 13, and 14 exhibits the PSNR comparisons, performance of feedback, and prediction for the "*all*" sequence when coded 64 kbps. The experiment used random delay of 0.02 seconds for the feedback packets, c was set to 1, the packet loss rate was 18.63%, and the values of T_1

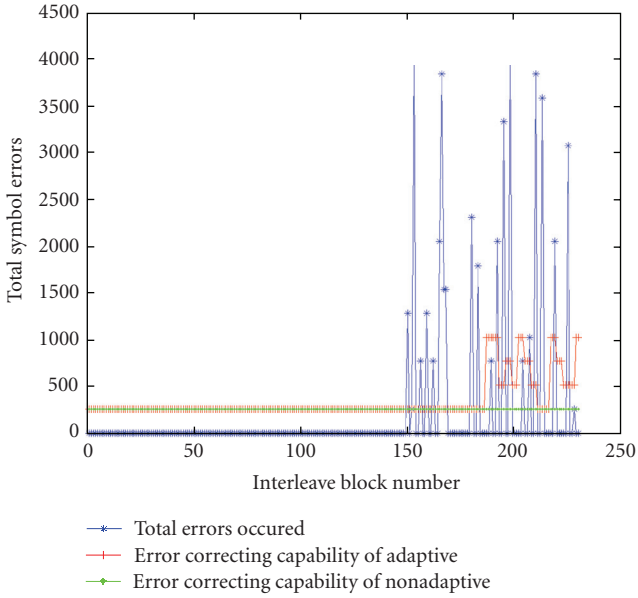


Figure 7: Performance of feedback for "*all*" sequence at 32 kbps, random delay 0.026, buffer size 500, packet size 256, span 16, PL rate 4.9%.

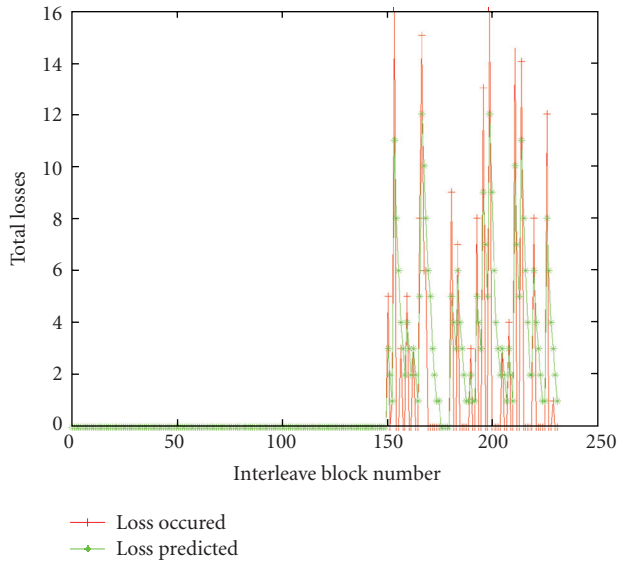


Figure 8: Performance of prediction for "*all*" sequence at 32 kbps, random delay 0.026, buffer size 500, packet size 256, span 16, PL rate 4.9%.

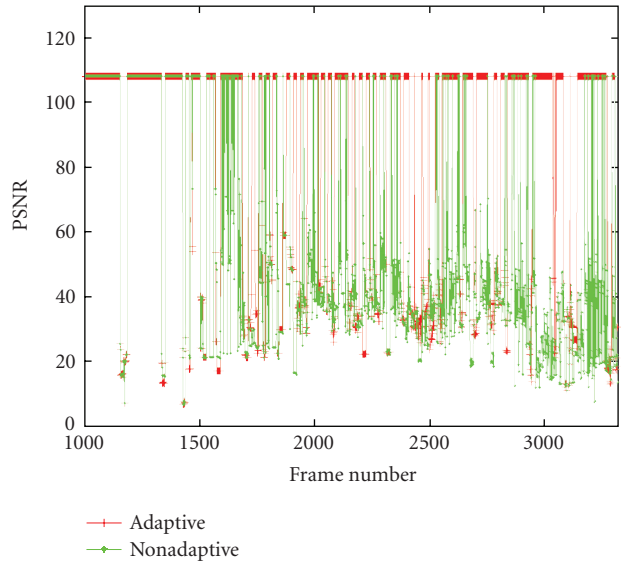


Figure 9: PSNR comparisons for "*all*" sequence at 32 kbps, random delay 0.028, buffer size 175, packet size 256, span 8, PL rate 10.07%.

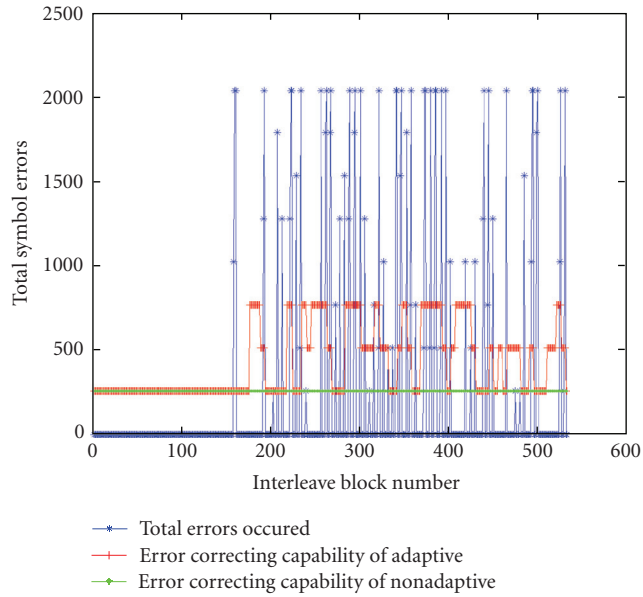


Figure 10: Performance of feedback for "*all*" sequence at 32 kbps, random delay 0.028, buffer size 175, packet size 256, span 8, PL rate 10.07%.

and T_2 used were 1 and 4 which are approximately 5% and 20% of the number of packets in the interleaving block. The PSNR performance was better in the adaptive case when compared to the nonadaptive case by about 31.1%. The adaptive algorithm had the capacity to correct the losses 18.80% of the time, whereas the nonadaptive algorithm had the capacity to correct the losses 6.84% of the time. The mean error in predicting the number of packets and hence symbols lost is 3.41 with a variance of 9.32.

Figures 15, 16, and 17 illustrate the PSNR comparisons, performance of feedback for "*all*" sequence at 128 kbps. The experiment used a packet loss rate of 15.66%, random delay of 0.012 seconds for the feedback packets, c was set to 1, and the values of T_1 and T_2 used were 1 and 3 which are approximately 5% and 20% of the number of packets in the interleaving block. The PSNR performance was better in the adaptive case when compared to the nonadaptive case by about 26.56%. The adaptive algorithm had the capacity

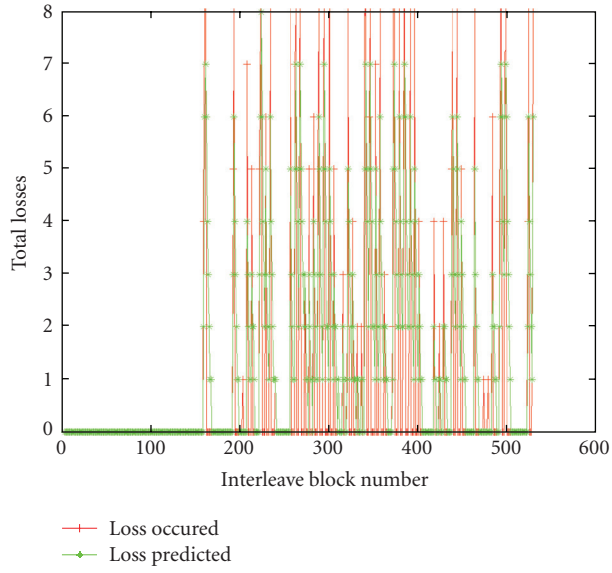


Figure 11: Performance of prediction for "*all*" sequence at 32 kbps, random delay 0.028, buffer size 175, packet size 256, span 8, PL rate 10.07%.

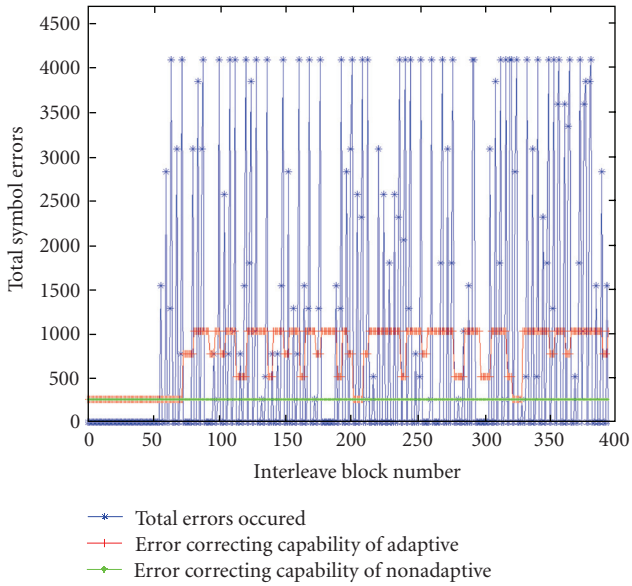


Figure 13: Performance of feedback for "*all*" sequence at 64 kbps, random delay 0.02, buffer size 250, packet size 256, span 16, PL rate 18.63%.

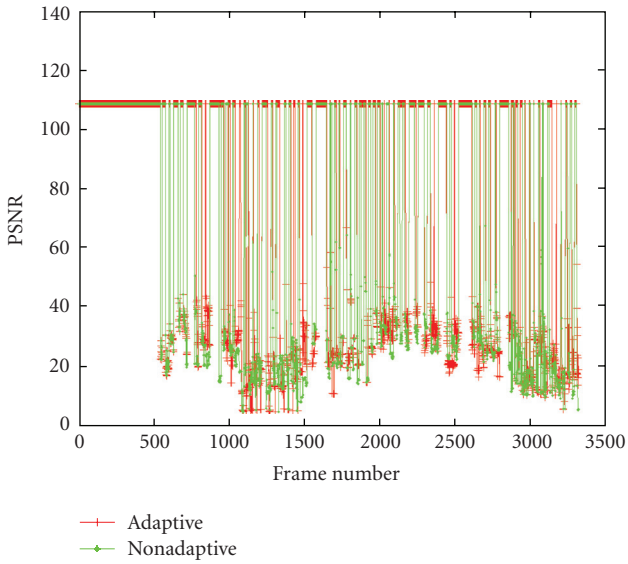


Figure 12: PSNR comparisons for "*all*" sequence at 64 kbps, random delay 0.02, buffer size 250, packet size 256, span 16, PL rate 18.63%.

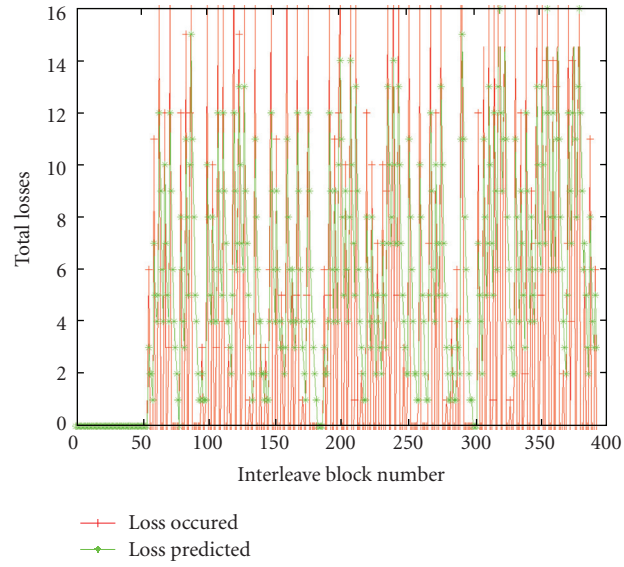


Figure 14: Performance of prediction for "*all*" sequence at 64 kbps, random delay 0.02, buffer size 250, packet size 256, span 16, PL rate 18.63%.

to correct the losses 10.53% of the time, in contrast to the nonadaptive algorithm which had the capacity to correct the losses 4.14% of the time. From Figure 17 it can be seen that the estimation was effective and resulted in an error in predicting the number of packets and hence symbols lost whose mean is 0.94 with a variance of 2.24. The results are summarized in Table 2.

Table 3 presents the utilization of the *all* sequences at different data rates and proxy parameters. BW (A-O) stands for the percentage increase in data rate between the original (O) and adaptive (A) cases, BW (A-NA) denotes the percentage increase in data rate between the adaptive (A) and nonadaptive (NA) data, and BW (NA-O) indicates the percentage increase in data rate between the nonadaptive (NA) and original (O) cases. As can be seen, the additional bandwidth used decreases as the span sizes increases. Based on the estimated packet loss, the server tries to change the message length K according to the rule N-TP/S. Since the code words are written column by column into the interleaving block and read out row by row, each packet

Table 2: Comparison of the performance for all at data rates 32, 64, and 128 kbps at packet losses of 4.9%, 10.07%, 18.63%, and 15.66%, respectively.

Sequence, rate (kbps), packet loss rate (%)	PSNR gain of adaptive over nonadaptive	Loss recovery of adaptive	Loss recovery of nonadaptive
all, 32, 4.9	12.54%	17.39%	8.70%
all, 32, 10.07	62.49%	16.88%	7.79%
all, 64, 18.63	31.10%	18.80%	6.84%
all, 128, 15.66	26.56%	10.53%	4.14%

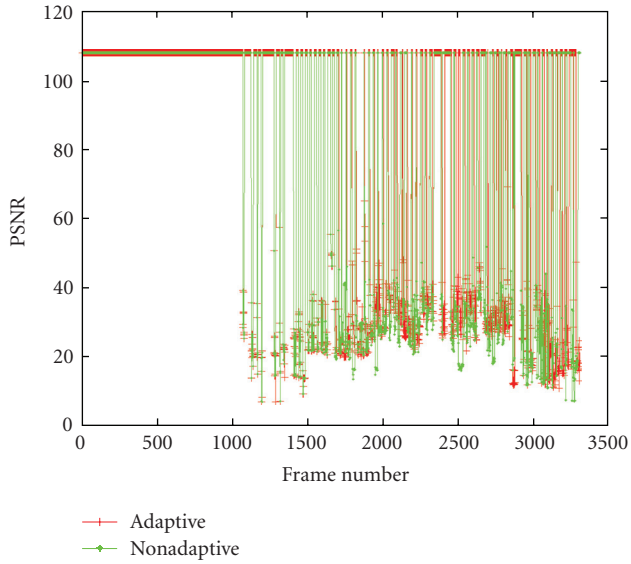


Figure 15: PSNR comparisons for "*all*" sequence at 128 kbps, random delay 0.012, buffer size 500, packet size 256, span 8, PL rate 15.66%.

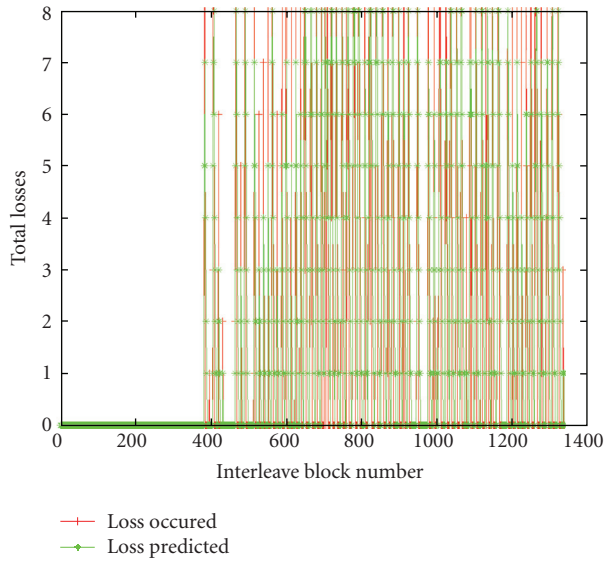


Figure 17: Performance of prediction for "*all*" sequence at 128 kbps, random delay 0.012, buffer size 500, packet size 256, span 8, PL rate 15.66%.

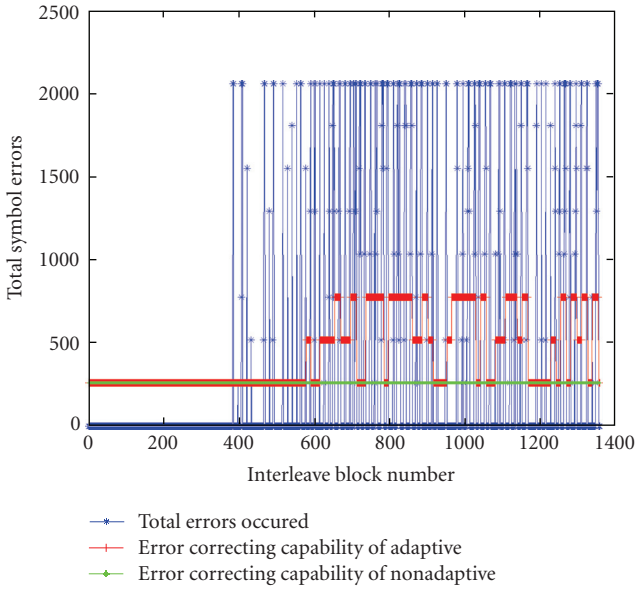


Figure 16: Performance of feedback for "*all*" sequence at 128 kbps, random delay 0.012, buffer size 500, packet size 256, span 8, PL rate 15.66%.

corresponds to P/S rows in the interleaving block. In order to correct a single packet, there should be P/S parity symbols, however, the value P/S decreases with an increase in span size. As the span size increases, the number of rows a packet occupies in the interleaved block decreases, and consequently the number of parity symbols used to correct a packet decreases. This would decrease the value of K indicating that for the same packet size less protection is required to recover a packet when the span size increases, which in turn results in less bandwidth utilization when the span sizes are increased.

Table 4 summarizes the performance of the adaptive error resilience strategy versus the nonadaptive error strategy for different data rates, delays, packet sizes, buffer sizes, span, and packet loss rates. The last column in Table 4 indicates the improvement in PSNR when using the adaptive versus the nonadaptive method. It can be observed that the adaptive algorithm works better at packet losses higher than 6%. In many cases the number of errors fixed by the adaptive strategy was twice as much as those fixed by the nonadaptive one. At lower packet losses, estimating future packet losses was not very effective, and the performance of both the adaptive and nonadaptive was close. It was also observed that the server was not able to effectively estimate

Table 3: Bandwidth utilization of QCIF sequences at different proxy parameters.

Sequence (bit rate in kbps)	Delay	Packet size	Buffer size	Span	Packet loss (%)	BW (A-O) (%)	BW (A-NA) (%)	BW (NA-O) (%)
all (32)	0.0260	256	500	16	4.90	13.11	2.22	10.65
all (32)	0.0280	256	175	8	10.07	26.35	10.42	14.44
all (32)	0.0350	128	175	16	16.57	14.95	3.88	10.65
all (32)	0.0155	128	500	8	17.54	29.17	12.60	14.44
all (64)	0.0209	256	600	8	6.11	21.97	3.01	18.40
all (64)	0.0200	128	600	16	10.21	19.05	7.62	10.61
all (64)	0.0200	256	250	16	18.63	27.98	15.24	10.61
all (64)	0.0200	256	250	8	14.86	32.00	11.50	18.40
all (128)	0.0200	256	500	8	10.45	27.22	7.40	18.44
all (128)	0.0135	128	1000	16	17.86	16.92	5.61	10.64
all (128)	0.0120	256	500	8	15.66	30.80	10.51	18.44
all (128)	0.0120	256	1000	16	5.25	13.32	2.44	10.62

Table 4: Performance comparisons for *all* at different rates and proxy settings.

Sequence (Bit rate in kbps)	Delay	Packet size	Buffer size (in packets)	Span	Packet loss (%)	Gain in PSNR (%)
all (32)	0.0260	256	500	16	4.90	12.54
all (32)	0.0280	256	175	8	10.07	62.49
all (32)	0.0350	128	175	16	16.57	37.34
all (32)	0.0155	128	500	8	17.54	37.92
all (64)	0.0209	256	600	8	6.11	16.48
all (64)	0.0200	128	600	16	10.21	30.84
all (64)	0.0200	256	250	16	18.63	31.1
all (64)	0.0200	256	250	8	14.86	20.32
all (128)	0.0200	256	500	8	10.45	23.6
all (128)	0.0135	128	1000	16	17.86	26.83
all (128)	0.0120	256	500	8	15.66	25.56
all (128)	0.0120	256	1000	16	5.25	6.34

sudden high losses that were preceded and followed by continuous low losses. Furthermore, it can be seen from Table 4 that as the packet sizes increased, the performance of both strategies decreased. This can be attributed to the following: as the packet sizes decreased, the threshold values T_1 and T_2 increased. Larger T_1 and T_2 values lead to better error recovery, as the server can change the value of K over a wide range depending on the range of T_2–T_1. Finally, it was observed that using $c = 1$ resulted in under prediction and using $c = 2$ resulted in over protection. This is because $c = 2$ results in larger estimate of error, and hence the server increases its protection levels in anticipation of many errors.

5. Conclusion

This paper described an adaptive system for error resilient transmission of compressed video over data networks. The system utilizes feedback information sent by the client to the server to enable the server to predict future channel conditions and consequently update the level of protection given to the data. The objective was to efficiently utilize bandwidth. It was observed that adapting the protection level performed much better than using nonadaptive forward error protection. The paper also described an emulation system used to emulate a data network. The system used three computers, one dedicated to be a server, another to serve as the client, and the third (the proxy) to play the role of a data network. The proxy was used to emulate randomly delayed data packets as well as to drop them in the event of buffer overflow.

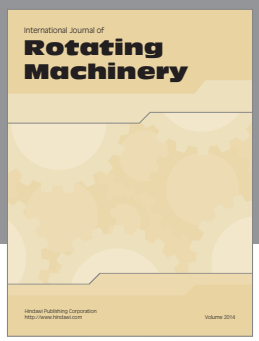
References

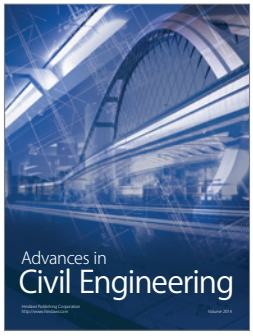
[1] G. J. Sullivan and T. Wiegand, "Video compression-from concepts to the H.264/AVC standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.

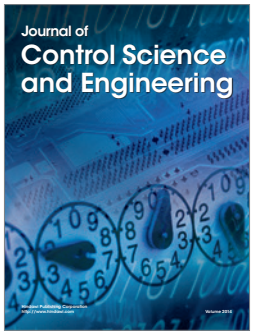
[2] J. L. Black, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, Chapman & Hall, New York, NY, USA, 1996.

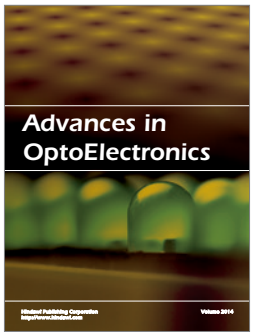
[3] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman & Hall, New York, NY, USA, 1997.

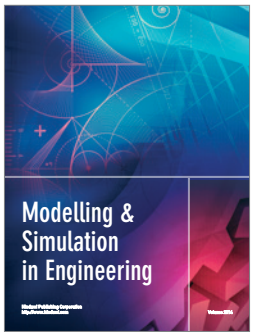
[4] R. Koenen, F. Pereira, and L. Chiariglione, "MPEG-4: context and objectives," *Signal Processing: Image Communication*, vol. 9, no. 4, pp. 295–304, 1997.
[5] ISO/IEC 144962-2, "Overview of the MPEG-4 Standard," ISO, 1997.
[6] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.
[7] ITU-T, "Draft ITU-T Recommendation H.263 Version 2: Video Coding for Low Bitrate Communication," International Telecommunication Union, 1997.
[8] G. Côté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, 1998.
[9] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.
[10] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.
[11] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Error-resilient video transmission using long-term memory motion-compensated prediction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1050–1062, 2000.
[12] T. Stockhammer, H. Jenkač, and C. Weiss, "Feedback and error protection strategies for wireless progressive video transmission," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 465–482, 2002.
[13] O. Harmanci and A. M. Tekalp, "A stochastic framework for rate-distortion optimized video coding over error-prone networks," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 684–697, 2007.
[14] Y. Zhang, W. Gao, Y. Lu, Q. Huang, and D. Zhao, "Joint source-channel rate-distortion optimization for H.264 video coding over error-prone networks," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 445–454, 2007.
[15] T. Stockhammer, H. Jenkač, and G. Kuhn, "Streaming video over variable bit-rate wireless channels," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 268–277, 2004.

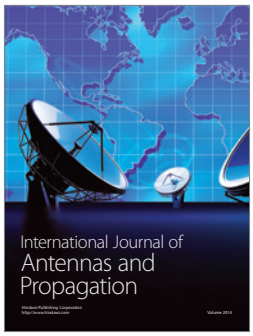


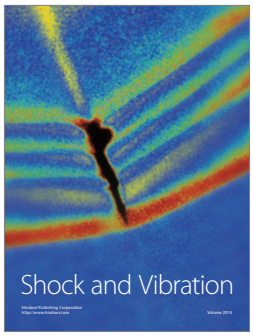


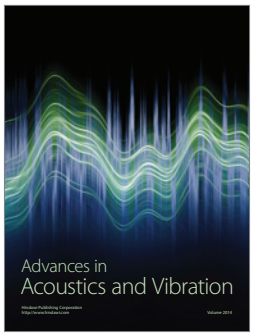












Hindawi

Submit your manuscripts at
http://www.hindawi.com