Efficient Rate Control for Video Streaming

Joseph C. Dagher, Ali Bilgin and Michael W. Marcellin

Dept. of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721

ABSTRACT

With the increasing importance of heterogeneous networks and time-varying communication channels, such as packet-switched networks and wireless communications, fine scalability has become a highly desirable feature in both image and video coders. A single highly scalable bitstream can provide precise rate control for constant bitrate (CBR) traffic and accurate quality control for variable bitrate (VBR) traffic. In this paper, we propose two methods that provide constant quality video under buffer constraints. These methods can be used with all scalable coders. Experimental results using the Motion JPEG2000 coder demonstrate substantial benefits.

Keywords: Video streaming, JPEG2000, Rate control, Bit allocation, Buffering, Leaky-bucket, Scalable.

1. INTRODUCTION

Streaming media over heterogeneous lossy networks and time-varying communication channels has become an active area of research. Several video coders that operate under the varying constraints of such environments have been recently proposed[1–5] . These coders rely on feedback control algorithms that estimate the available bandwidth in the network. The controllers drive the encoding algorithm and adjust its rate accordingly, thus providing a higher average PSNR with less skipped frames, and decreasing the standard deviation of the PSNR values.

Recently, scalability has become a very desirable feature in video coding. Besides several other advantages, scalable video coders produce excellent results when they are coupled with efficient rate control (RC) algorithms. Some video coders offer scalability on a coarse level, such as the MPEG-2 and H.263 coders that produce layered bitstreams. Others offer fine scalability where the bitstreams can be decoded at any bitrate up to and including a maximum. Various attempts that make use of both fine and coarse scalability for efficient rate allocation have appeared in the literature. In [6,7] the authors utilize scalable codecs to achieve constant quality video. Scalable codecs were also used in [8–10] to adaptively accommodate changing network conditions.

In this paper, we propose two leaky-bucket rate allocation methods that provide constant quality video under buffer constraints. The basic block diagram of the proposed methods is given in Figure 1. This strategy relies on the highly scalable nature of JPEG2000, and can be summarized as follows: As illustrated in Figure 1, each frame is individually encoded using JPEG2000 at a rate somewhat above the target rate for the sequence. The compressed bitstreams are placed in a buffer prior to transmission or storage. The data is pulled out of the buffer at a constant rate. If the buffer is full (or nearly full) when a new compressed frame arrives, the bitstreams that are already in the buffer are truncated via the embedding property to maintain constant quality across all frames in the buffer.

This paper is organized as follows. An overview of JPEG2000 and Motion JPEG2000 is presented in Section 2. Section 3 presents the two algorithms proposed for efficient video streaming. Experimental results are presented in Section 4, while Section 5 presents concluding statements.

2. OVERVIEW OF JPEG2000 AND MOTION JPEG2000

JPEG2000 is the latest ISO/IEC image compression standard. Here, we will provide a high level description of the JPEG2000 algorithm to assist the reader in comprehending the remainder of this paper. For a more thorough description, the interested reader is referred to [11–13] .

A simplified block diagram of a JPEG2000 encoder is illustrated in Figure 2. The input image is first passed through an optional component transform to achieve decorrelation across color components. The resultant components are wavelet transformed and quantized. Each subband is then divided into codeblocks. Codeblocks are

Further author information: (Send correspondence to J.C.D.) J.C.D.: joseph@ece.arizona.edu, A.B.: bilgin@ieee.org, M.W.M.: marcellin@ece.arizona.edu

Applications of Digital Image Processing XXIV, Andrew G. Tescher, Editor, Proceedings of SPIE Vol. 4472 (2001) © 2001 SPIE · 0277-786X/01/$15.00

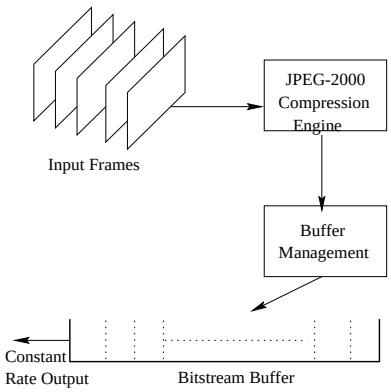

Figure 1. Basic block diagram of the proposed algorithm.

compressed independently using a bitplane coder. The bitplane coder makes three passes over each bitplane of a codeblock. Each of these passes are referred to as coding passes or subbitplanes. Thus, an embedded bitstream is generated for each codeblock. The JPEG2000 encoder computes and stores the rate-distortion information corresponding to each subbitplane of every block.

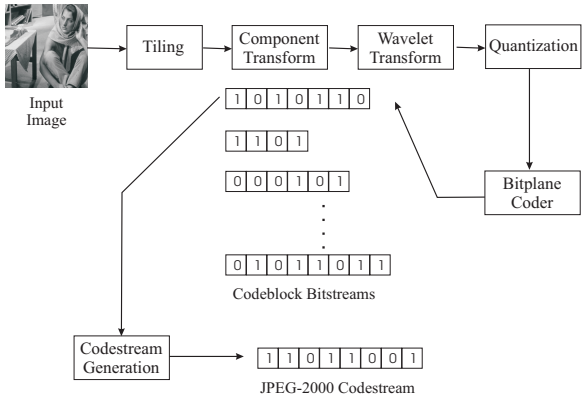

Figure 2. Block diagram of a JPEG2000 encoder.

The creation of a JPEG2000 codestream involves the inclusion of a different number of coding passes from each individual codeblock bitstream. JPEG2000 offers tremendous flexibility in this regard. The decision on how many coding passes of a particular codeblock bitstream should be included can be based on any desired criteria. For example, optimum rate-distortion performance at a given target rate is achieved when the coding passes with greatest distortion-rate slopes are included.

JPEG2000 includes the following features:

• Superior compression performance: JPEG2000 provides excellent compression performance compared to previous standards; especially at low rates.

• Multi-component image compression: JPEG2000 can handle images with dynamic ranges varying from 1 bit to 16 bits for each color component.

• Lossless and lossy compression can be obtained from one bitstream in the course of progressive decoding.

• Progressive transmission by pixel accuracy and resolution that allows the reconstruction of images at any rate and various resolution levels.

• Random code-stream access and processing to allow operations such as compressed domain cropping, rotation, translation, filtering, feature extraction, scaling, etc.

• Region-Of-Interest (ROI) encoding/decoding.

• Robustness to bit-errors.

The JPEG committee has decided to extend the JPEG2000 standardization effort to video coding. A work item, referred to as Motion JPEG2000 (MJP2), has been approved and is currently under development. MJP2 is essentially a file format for wrapping compressed frames generated by the JPEG2000 image coding engine [14–16]. It is intended to generate a highly scalable bitstream, which can be easily edited. Thus, MJP2 does not include motion compensation. Each frame is individually compressed and stored. The scope of MJP2 encompasses video compression for applications including Digital Still Camera (DSC) and Camcorder, remote surveillance systems, digital video recording systems, and video capture cards. Preliminary results indicate that substantial performance gain and functionality can be achieved over existing Motion-JPEG methods [16]. In a later section, we will analyze trade-offs between decoded video quality, buffer size, and delay.

3. THE LEAKY-BUCKET ALGORITHM

The goal of the work described in this paper was to devise an algorithm to achieve constant quality video under buffer and rate constraints. Let N denote the number of frames to be encoded and let R denote the average rate per pixel, per frame. Thus the total bit budget for encoding all N frames is NR. Let D_i and $R_i, i \in \{1, 2, \ldots, N\}$, denote the distortion and rate associated with the ith frame, respectively. Let B denote the size of the buffer that is used to hold the compressed image sequence.

For a given buffer size B, the problem is to achieve minimum average distortion under the constraint that the total bit budget is not exceeded. In other words, for a fixed B, we would like to select R_i such that

$$\underset{R_i}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} D_i \tag{1}$$

subject to the constraint that

$$\sum_{i=1}^{N} R_i = NR. \tag{2}$$

The solution to this problem is given by [17]

$$R_i = R + \frac{1}{2} \log_2 \frac{\sigma_i^2}{G}. \tag{3}$$

when the corresponding distortions are modeled by

$$D_i = G\epsilon^2 2^{-2R}, \tag{4}$$

where ϵ^2 is a constant that takes into account the performance of practical quantizers and G is the geometric mean of the variances of the frames, $\sigma_i^2, i \in \{1, 2, \ldots, N\}$, given by

$$G = \left[\prod_{i=1}^{N} \sigma_i^2 \right]^{\frac{1}{N}}. \tag{5}$$

It can be seen from Equation (4) that D_i is constant, $\forall i \in \{1, 2, \ldots, N\}$. This suggests that for the simple model employed here, minimizing the average distortion should result in individual distortions being equal across all frames. In other words, minimizing average distortion should result in constant quality, as desired.

It is important to point out two extreme cases at this point. The first one is when the buffer size is equal to the size of the entire compressed sequence, i.e. $B = NR$. This will clearly yield the best result, however for large N, buffering the entire compressed sequence may not be feasible due to memory constraints. Furthermore this approach will result in very large latency. The other extreme case is when only a single compressed frame is buffered. This case will provide minimum latency. However, the quality of the decoded sequence will vary widely across frames depending on rate-distortion properties of the sequence.

The algorithm presented in this paper was motivated by the work of [18] which presents a low memory implementation of a JPEG2000 image coder for coding a single frame. That algorithm employs a sliding window wavelet transform to generate wavelet coefficients in an incremental fashion. Each time enough lines of wavelet coefficients are available, they are divided into codeblocks, quantized, and entropy coded. The resulting embedded block bitstreams are subsequently sent to an output (FIFO) buffer. Compressed data are removed from this buffer for transmission at a constant rate. Rate allocation is implicitly performed through the algorithm by which compressed data are added to the buffer.

Whenever such data are to be added to the buffer, there is a possibility that not enough buffer space is available. When this occurs, the coding passes having lowest distortion-rate slopes are discarded. In general, these discarded coding passes come from both the buffer and the newly compressed data that is to be added to the buffer.

3.1. Single buffer rate controller (SBRC)

In what follows we will describe what is an extension to the work of [18] . As stated earlier, our MJP2 rate controller will be implemented to provide constant decoded video quality subject to buffer constraints. We have implemented two different RC algorithms to accomplish this task.

Figure 1 shows a basic block diagram of our proposed SBRC algorithm. As shown, each frame is compressed independently using the JPEG2000 coding engine. The compression rate of each frame is somewhat greater than the target rate for the sequence. The resulting compressed bitstream is placed in a buffer awaiting transmission or storage. Then, the data is pulled out of the buffer at a constant rate. When the buffer is (or about to be) full, all bitstreams, including the ones already in the buffer along with the new bitstream to be inserted, are truncated via the embedding property to maintain constant quality across all frames in the buffer. This strategy relies on the highly scalable nature of JPEG2000. The SBRC algorithm uses a single RC buffer to achieve constant quality. The algorithm is described in more detail in Table 3.1.

3.2. Double buffer rate controller (DBRC)

Although the SBRC algorithm performs reasonably well under most conditions, it is possible to improve its performance. To see this, consider the scenario where we have $M-1$ frames already in the RC buffer. Furthermore, assume that the coding passes of those frames have been truncated according to a RD threshold of T_1. Suppose that the next frame to be inserted in the buffer is such that most of its coding passes have RD slopes smaller than T_2, where $T_2 \ll T_1$. As a result, the new RD threshold T_{RD} computed for all M frames will be $T_{RD} < T_1$. However, having permanently truncated the coding passes of the first $M-1$ frames with RD slopes less than T_1, we will be obliged to include the coding passes with RD slopes less than T_2 from the new frame, or allow the buffer to remain at less than full occupancy. In this situation, it is desirable to be able to "reclaim" coding passes discarded from other frames in the buffer.

To this end, we introduce the DBRC algorithm where some of the coding passes that have been eliminated in previous iterations are kept in a secondary buffer of predetermined size. The DBRC algorithm allows these coding passes to be considered again at a later stage. It should be noted that once a frame is released, all of its passes residing in the secondary buffer will be permanently thrown away. The DBRC algorithm is described in Table 3.2.

4. EXPERIMENTAL RESULTS

In this section we evaluate the performance of the SBRC and DBRC algorithms on individual video sequences. We use Peak Signal-to-Noise-Ratio (PSNR) as the distortion measure. Figure 3 illustrates the performance of the SBRC algorithm on the 8-bit grayscale Susie sequence. Each frame has 705×480 pixels. Four different buffer sizes were used in the figure. First the buffer size B was set to include the entire sequence of 150 frames at the target rate of 1.0 bits/pixel, i.e. $B = 150 \times 1.0 \times 705 \times 480$ bits. As discussed earlier, this case results in minimum variance for PSNR values amongst all other buffer sizes, under the same constraints. Unfortunately, it also results in maximum latency and has the largest memory requirements. The other extreme case is when the buffer size is set to the size of a single frame compressed at the target rate. This case has the least amount of latency. However, it produces the largest variance for PSNR values. Results using buffer sizes that correspond to 30 and 45 frames are also included in the figure. It can be seen that buffering 30 and 45 compressed frames reduces the variance of the PSNR by 45% and 60%, respectively, compared to buffering only a single compressed frame.

Given the size of the RC buffer B in bytes, determine the number of frames that will fit in the buffer, M, using $M = \frac{BD}{SR}$, where S is the size of one frame in pixels, R is the desired bit rate in bits/pixel, and D is the pixel bit-depth.

Determine an RD threshold T_{RD} such that the coding passes of the first frame with RD slopes $\geq T_{RD}$ will fit into the buffer.

Delete the coding passes of the first frame with RD slopes less than T_{RD}.

for $k = 2$ to $k = N + M$

if $k \leq N$

Determine T_{RD} so that coding passes of the frames currently in the buffer and those of the kth frame with slopes $\geq T_{RD}$ will fit in the buffer.

Delete the coding passes of the frames currently in the buffer and those of the kth frame with RD slopes $< T_{RD}$.

Insert the qualifying coding passes of the kth frame into the buffer.

end if

if $k > M$

Release SR bits from the head of the buffer to the codestream.

end if

Set $k = k + 1$

end for

Table 1. SBRC Algorithm.

Figure 4 shows the performance of the DBRC algorithm at the same target rate of 1.0 bits/pixel. The buffer ratio used in this figure is defined as

$$\text{buffer ratio} = \frac{\text{primary buffer size} + \text{secondary buffer size}}{\text{primary buffer size}}. \tag{6}$$

A buffer ratio of 1.5 was used to obtain the results in Figure 4. This means that the size of the secondary buffer was half the size of the primary buffer. By comparison of Figures 3 and 4, it can be seen that the DBRC algorithm provides substantial improvements over the SBRC algorithm. For a buffer size of 30 frames, using DBRC with a buffer ratio of 1.5 decreases the variance of the PSNR by 40% compared to the SBRC algorithm. This reduction rises to 62% for a buffer size of 45 frames. Thus, for a fixed amount of latency, DBRC provides significant reduction in variance compared to SBRC. To compare the performance of the two algorithms for fixed memory consumption, we examine the SBRC algorithm with a buffer size corresponding to 45 frames and the DBRC algorithm with a buffer size corresponding to 30 frames and a buffer ratio of 1.5. In this case, the DBRC provides a 20% reduction in variance over SBRC, while introducing only 66% of the latency of the SBRC algorithm.

It has been observed that increasing the size of the secondary buffer, or equivalently, the buffer ratio improves performance. However, there is a saturation point over which the results do not improve significantly. This is expected. Recall that the secondary buffer temporarily stores those coding passes whose RD slopes are too low to make it to the primary buffer but high enough to be kept for future consideration. After a certain point, the secondary buffer will start holding coding passes which have very low chance of making it to the primary buffer. This is where saturation occurs. This is illustrated in Figure 5. In this figure, results obtained using the DBRC algorithm

Given the size of the primary RC buffer B^p in bytes, determine the number of frames that will fit in the buffer, M, using $M = \frac{B^p D}{SR}$, where S is the size of one frame in pixels, R is the desired bit rate in bits/pixel, and D is the pixel bit-depth.

Determine the primary RD threshold T^p_{RD} such that the coding passes of the first frame with RD slopes $\geq T^p_{RD}$ will fit into the primary buffer.

Determine the secondary RD threshold T^s_{RD} such that the remaining coding passes of the first frame with RD slopes $\geq T^s_{RD}$ will fit into the secondary buffer.

Delete the coding passes of the first frame with RD slopes less than T^s_{RD}.

for $k = 2$ to $k = N + M$

if $k \leq N$

Determine T^p_{RD} so that coding passes of the frames currently in the buffer and those of the kth frame with slopes $\geq T^p_{RD}$ will fit in the primary buffer.

Determine T^s_{RD} so that remaining coding passes of the frames currently in the buffer and those of the kth frame with slopes $\geq T^s_{RD}$ will fit in the secondary buffer.

Delete the coding passes of the frames currently in the buffer and those of the kth frame with RD slopes $< T^s_{RD}$.

Insert the qualifying coding passes of the kth frame into the primary and secondary buffers.

end if

if $k > M$

Release SR bits from the head of the primary buffer to the codestream.

end if

set $k = k + 1$

end for

Table 2. DBRC Algorithm.

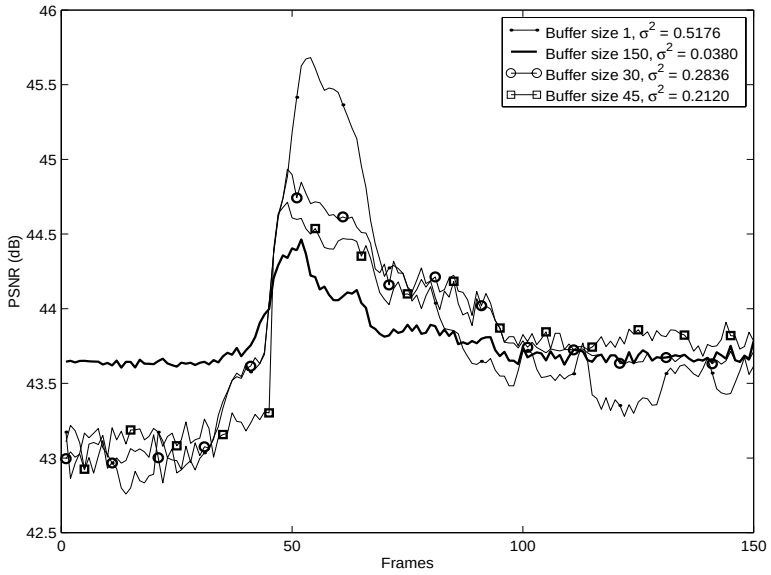

Figure 3. Performance of the SBRC algorithm on the Susie sequence encoded at an average rate of 1.0 bits/pixel.

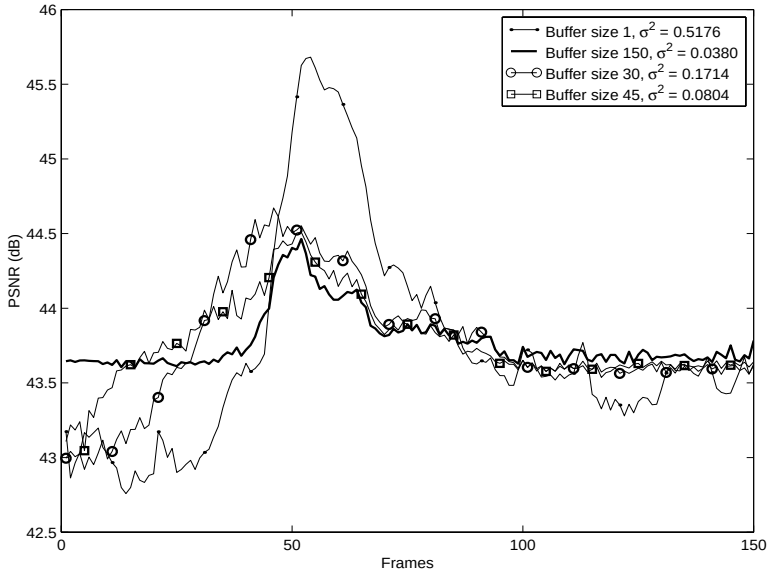

Figure 4. The performance of the DBRC algorithm on the Susie sequence encoded at an average rate of 1.0 bits/pixel.

with different buffer ratios on the Susie sequence are plotted. The buffer ratios vary between 1.0 and 1.75. It can be seen that the performance reaches a saturation point around a buffer ratio of 1.5.

The performance of DBRC is especially impressive when sudden jumps occur in RD slopes of successive frames. This is the case when scene changes occur in a video sequence. To illustrate this, we present the performance of the DBRC algorithm with a buffer ratio of 1.5 using the 150 frame QCIF sequence Trevor, which has a scene change at frame 60. Figure 6 illustrates the PSNR values for several different buffer sizes for a rate of 1.0 bits/pixel. The figure compares the performances of five different buffer sizes, 1, 20, 40, 60, and 150. Compared to the single frame buffer size case (with a buffer ratio of 1.5), the variances of the PSNR values for buffer sizes corresponding to 20, 40, and 60 frames decreased by 27%, 55%, and 79%, respectively. It should be noted that, throughout all experiments, our proposed rate control algorithms had very little effect on the average PSNR of the video sequence. When any significant difference does occur, larger buffer sizes are favored.

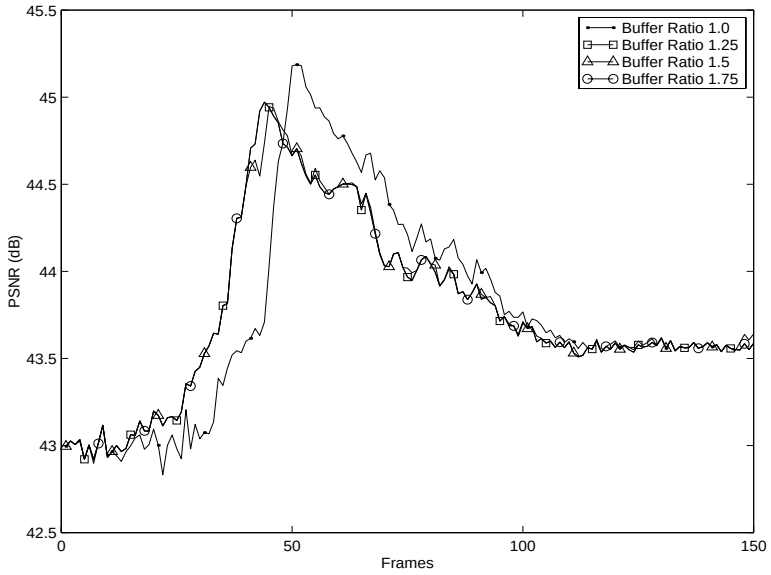

Figure 5. The performance of the DBRC algorithm with a primary buffer size of 20 on the Susie sequence.

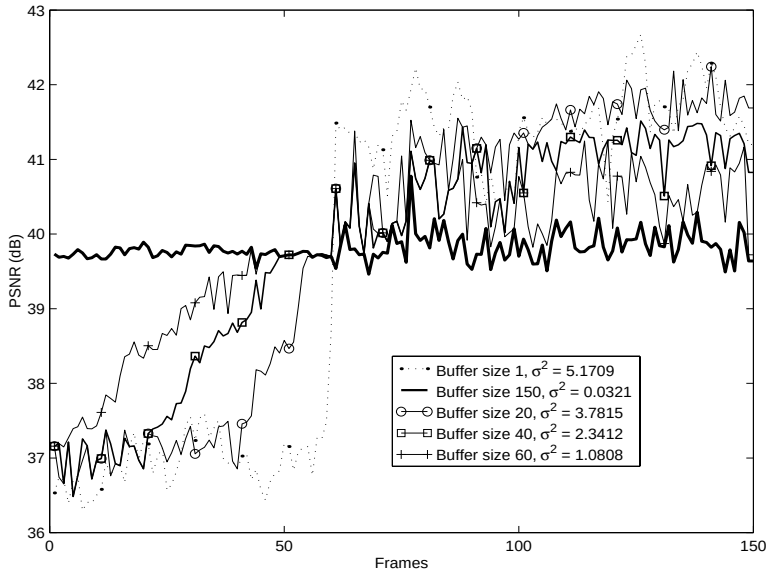

Figure 6. The performance of the DBRC algorithm on the Trevor sequence.

5. CONCLUSIONS

In this paper, we present two new rate control algorithms, SBRC and DBRC, that can be applied to any compression scheme capable of fine scalability. The algorithms significantly reduce the quality fluctuations among frames, and provide smoother video sequences. When used with the upcoming Motion JPEG2000 standard, both algorithms produce conformant bitstreams. Simulations show that the proposed algorithms adapt well to varying conditions.

REFERENCES

1. Q. Zhang, W. Zhu, and Y. Zhang, "Network-adaptive rate control with TCP-friendly protocol for multiple video objects," in *IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 1055 –1058, 2000.
2. Y. Hou, W. Dapeng, Z. Wenwu, J. Hung, C. Tihao, and Y. Zhang, "An end-to-end architecture for MPEG-4

video streaming over the internet," in *IEEE International Conference on Image Processing*, vol. 1, pp. 254–257, 1999.
3. C. Fung and S. Liew, "End-to-end frame-rate adaptive streaming of video data," in *IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 67–71, 1999.
4. Y. Chung, H. Song, T. Kuo, J. Kim, and C. J. Kuo, "Continually traffic accommodating internet streaming video," in *IEEE Second Workshop on Multimedia Signal Processing*, pp. 403–408, 1998.
5. I. Pao and M. Sun, "Encoding stored video for streaming applications," *IEEE Transactions on Circuits and Systems for Video Technology* **11**, pp. 199–209, Feb. 2001.
6. H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable internet video using MPEG-4," *Signal Processing: Image Communication* **15**, pp. 95–126, Sept. 1999.
7. M. van-der Schaar and H. Radha, "A hybrid temporal-SNR fine-granular scalability for internet video," *IEEE Transactions on Circuits and Systems for Video Technology* **3**, pp. 318–331, 2001.
8. J. Rexford, S. Sen, and A. Basso, "A smoothing proxy service for variable-bit-rate streaming video," in *Global Telecommunications Conference-GLOBECOM'99*, vol. 3, pp. 1823–1829, 1999.
9. D. Reininger, M. Ott, G. Michelitsch, and G. Welling, "Scalable QoS control for VBR video servers," in *IEEE First Workshop on Multimedia Signal Processing*, pp. 570–575, 1997.
10. E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and implementation of a caching system for streaming media over the internet," in *Sixth IEEE Real-Time Technology and Applications Symposium*, pp. 111–121, 2000.
11. "JPEG 2000 Part I Final Draft International Standard," *ISO/IEC JTC 1/SC 29/ WG1, Doc. No. N1855* , Aug. 2000.
12. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Practice and Standards*, Kluwer Academic Publishers, Massachusetts, 2001.
13. M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG-2000," in *Data Compression Conference*, pp. 523–541, Mar. 2000.
14. "Motion JPEG2000 (MJP2) requirements and profiles version 6.0," *ISO/IEC JTC 1/SC 29/ WG1, Doc. No. N2106* , Mar. 2001.
15. T. Fukuhara and D. Singer, "Motion JPEG2000 verification model ver.4.0 (technical description)," *ISO/IEC JTC 1/SC 29/ WG1, Doc. No. N1983* , Jan. 2001.
16. T. Fukuhara, "Presentation on Motion JPEG-2000," *ISO/IEC JTC 1/SC 29/ WG1, Doc. No. N1389* , July 1999.
17. N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, New Jersey, 1984.
18. T. Flohr, M. Marcellin, and J. Rountree, "Scan-based processing with JPEG-2000," in *Applications of Digital Image Processing XXIII, Proc. of SPIE*, vol. 4115, July 2000.