JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENT
Vol. 9, No. 3, September, pp. 211–222, 1998
ARTICLE NO. VC980389

Streaming Video Using Dynamic Rate Shaping and TCP Congestion Control†

Stephen Jacobs* and Alexandros Eleftheriadis*

Department of Electrical Engineering and Columbia New Media Technology Center, Columbia University, New York, New York 10027

Received February 9, 1998; accepted May 26, 1998

We present a new technique for streaming real time video on today's Internet, based on *dynamic rate shaping* and *TCP congestion control.* Dynamic rate shaping is a signal processing technique that adapts the rate of compressed video (MPEG-1, MPEG-2, H.26x) to dynamically varying bandwidth constraints. This provides an interface (or filter) between the source and the network, with which the encoder's output (either live or stored) can be perfectly matched to the network's available bandwidth. We couple this adaptation capability with the use of a new semi-reliable protocol that uses the TCP congestion window to pace the delivery of data into the network, but without using other TCP algorithms that are poorly suited to real time media. Use of TCP congestion control ensures that the protocol competes fairly with all other TCP data and that it optimally shares the available bandwidth. It also avoids the latency problems commonly associated with TCP. In addition, we describe a real application that uses this approach to stream MPEG video on the Internet. We present several experiments, performed in both a controlled environment and the wide area Internet, that were used to evaluate the effectiveness and fairness of the scheme. The results show that the proposed solution achieves superior video quality while at the same time providing fairness by sharing bandwidth equally with other non-real-time connections.

INTRODUCTION

The transmission of digital audiovisual information across a communication system is a well-understood problem. Its core lies in key assumptions made by codec designers about possible networks that may carry the compressed information and assumptions made by network designers about the types of traffic their network is intended to carry. Traditionally, codec design has assumed ideal networks (constant bandwidth and fixed delay), so that the semantics of synchronization and clock recovery can be unambiguously defined. The fixed delay assumption is allowed to be relaxed in practice, with the addition of extra buffering at the receiver. It is much more difficult to deviate from the constant bandwidth requirement; solutions from the codec side include multiresolution or scalable coding, as well as built-in robustness measures that aid in error recovery and concealment. From a networking standpoint, an approximation to such an ideal communication system can be provided using quality of service (QoS) guarantees. Such guarantees can aid in fully or partially characterizing network "imperfections," and allow the implementation of proactive measures to work around them.

Our interest in this paper is focused on Internet-based delivery of real-time digital video, and in particular MPEG-1 [14, 19] and MPEG-2 video [13, 15]. Both obviously represent the most pervasive technical solutions in their respective fields, and a successful solution that caters to their particular needs can have a significant impact.

The underlying technologies of today's Internet are not sufficient to support QoS guarantees which would facilitate real-time services. The evolution of these technologies could result in any number of possibilities, including ATM backbones, IP switching, or fully deployed ATM networks. Regardless of the specifics, it is likely that the network infrastructure of the future will have QoS and that users will be able to request connections with or without QoS. The exact horizon on when such capabilities will be widely available is, however, not clear. Connections which reserve resources will demand a higher cost; users may not always want to pay this extra cost for a particular real time service (e.g., watching movies vs video database browsing). In addition, parts of the network infrastructure may never be able to provide QoS, such as wireless connections, or may provide partial QoS, such as ATM-ABR that only provides a guaranteed minimum QoS. These considerations indicate that transmission of real-time information without QoS is not only important today, but is going to remain so even in networks that *can* provide QoS.

Lack of QoS translates into potential variation of the bandwidth available for video (and audio) transmission.

* E-mail: {sej,eleft}@ee.columbia.edu.

† This article is part of a special section devoted to the Image Technology for World-Wide-Web Applications.

1047-3203/98 $25.00

This variation can be quite unpredictable, both in terms of its short and long-term behavior. Although techniques have been developed to employ rate control for live sources based on network feedback [12, 16], these will not work with prerecorded material. In addition, MPEG-1 and MPEG-2 (hereafter referred to simply as MPEG) sources are typically coded at a fixed bit rate.[1] It is crucial then to be able to modify the bit rate of MPEG video, even after encoding has already taken place.

We refer to this rate manipulation operation as *dynamic rate sharing* (DRS) [8, 10, 11]. The term dynamic refers to the possibility that rate constraints are time-varying, while shaping is used instead of rate control to: (1) differentiate from classical encoder rate control in which the variable rate of an entropy-coded bitstream is matched to a fixed channel rate and (2) to more accurately capture the posterior (with respect to coding) nature of the operation. Note that DRS is quite different from traffic shaping (e.g., in DRS the traffic's average rate can change). In order for rate shaping to be viable, it has to be implementable with reasonable complexity (preferably in software) and yield acceptable visual quality. We present in this paper a summary of our work in this area [8, 10, 11], in which we have fully characterized optimal DRS and have identified extremely fast algorithms (much faster than an MPEG decoder) that perform within 0.5 dB of the optimal one.

In order to use the DRS approach in the environment presented by today's Internet, the key concern is the bit rate which information should be transmitted. The delay variation (jitter) can be mitigated by adapting the receiver's buffering, as we explain in more detail later. We have designed a transport protocol, based on TCP congestion control, with the main goal of facilitating unicast real time services that do not degrade the performance of other data transfers on the Internet, such as FTP, HTTP, etc. Since the Internet is a shared medium, there is always competition among users for its resources. The most important data transfer protocol is TCP, not only because it is dominant in sheer traffic volume, but also because it promotes fairness between different data transfers by sharing the available bandwidth evenly among users. Many real-time video streaming software packages that exist today make little or no effort at maintaining this fairness. We will show that our system preserves the fairness that the Internet and TCP originally intended.

Although our protocol was not intended to scale to multicast, we believe that unicast is a very important class of applications. There is a large body of work in the area of Internet streaming with rate controllable sources, although much of it has focused on multicast [3, 5–7, 16, 23]. Since unicast can be viewed as a special case of multicast, these protocols are often used for unicast. However, this is not the most efficient solution. Multicast protocols attempt to solve a superset of the problems posed by unicast. In this way, they contain trade-offs which may solve multicast problems, but at the expense of unicast performance (for example, they do not provide for fair sharing of bandwidth with traditional data sources).

[1] All commercial implementations of encoding chips or chip sets that we know of do not provide VBR output.

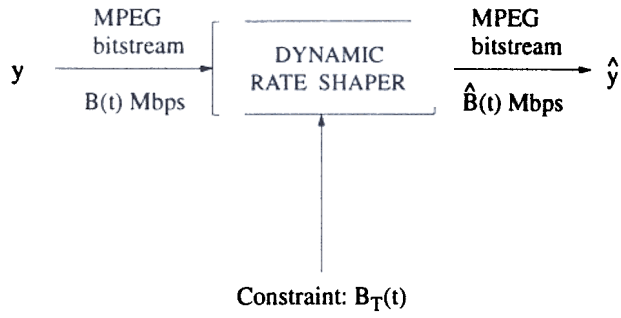

FIG. Operation of a dynamic rate shaper.

In the next section, we present an overview of DRS, including characteristic performance results. Section 3 describes the design of our transport protocol. In Section 4, we demonstrate a real application that uses this protocol to stream MPEG video, while sharing available bandwidth fairly. The applications consists of a Unix server and a Windows 95/NT client developed using the Microsoft ActiveMovie (recently renamed DirectShow) framework. We also present experimental results from a controlled environment and results from testing in the wide area Internet, across more than 30 hops. Finally, we present some concluding remarks in Section 5.

2. DYNAMIC RATE SHAPING

We define rate shaping as an operation which, given an input video bitstream and a set of rate constraints, produces a video bitstream that complies with these constraints. If the rate constraints are allowed to vary with time, the operation will be called dynamic rate shaping. For our purposes, both bitstreams are assumed to meet the same syntax specification. In this paper, we focus on MPEG-1 and MPEG-2, but we should point out that the technique applies to essentially any block-based transform coding scheme [8] (including H.26x and so-called "motion" JPEG). We assume that the reader is familiar with MPEG's main characteristics; overviews can be found in [13, 19], while the actual standards are detailed in [14, 15].

2.1. *DRS Problem Definition*

The rate shaping operation is depicted in Fig. 1. Note that no communication path exists between the rate shaper and the source of the input bitstream, which ensures that

no access to the encoder is necessary. The source of the rate constraints $B_T(t)$ for the purposes of this paper is the bandwidth availability estimate provided by our transport protocol, as described later on. The objective of a rate shaping algorithm is to minimize the conversion distortion, i.e.:

$$\min_{\hat{B}(t)\le B_T(t)} \{\|y - \hat{y}\|\}. \quad (1)$$

Note that no assumption is made on the rate properties of the input bitstream, which can indeed be arbitrary. In practice, it is typically a constant bit rate stream. The attainable rate variation ($\hat{B}/B$) is limited and depends primarily on the number of B pictures of the bitstream and the original rate $B(t)$.

Assuming that MPEG-2 (or, more generally, a motion-compensated block-based transform coding technique) is used to generate the input bitstream and decode the output one, there are two fundamental ways to reduce the rate: (1) modifying the quantized transform coefficients by employing coarser quantization and (2) eliminating transform coefficients. In general, both schemes can be used to perform rate shaping; requantization, however, leads to recoding-like algorithms which are not amenable to fast implementation and do not perform as well as selective-transmission ones. Consequently, we only consider selective-transmission-based algorithms, and more specifically we address the particular case of truncation (a set of DCT coefficients at the end of each block is eliminated). This approach will be referred to as *constrained* dynamic rate shaping. The more general case of *generalized* rate shaping where elimination of arbitrary DCT coefficients is allowed is discussed in [8, 11], where it is shown that it provides marginal quality improvement at significant additional computational complexity.

The number of DCT run-length codes within each block which will be kept will be called the *breakpoint* (similar to the breakpoint used in MPEG-2 data partitioning [12]). Assuming use of MPEG and avoiding certain syntax complications,[2] we require that at least one DCT coefficient will remain in each block. Consequently, breakpoint values will range from 1 to 64.

2.2. *Rate Shaping of Intra-Coded Pictures*

In intra-picture rate shaping, there is no temporal dependence between pictures. Consequently, the shaping error will simply consist of the DCT coefficients that are dropped. It can then be easily shown that the DRS problem can be expressed as

[2] These include recoding the coded block patterns and reexecuting DC prediction loops.

$$\min_{\hat{B}(t)\le B_T(t)} \{\|y - \hat{y}\|\} \Leftrightarrow \min_{\sum_{i=1}^{N} R_i(B_i)\le B_T} \left\{\sum_{i=1}^{N} D_i(b_i)\right\} \quad (2)$$

with

$$D_i(b_i) \equiv \sum_{k\ge b_i} [E^i(k)]^2 \quad (3)$$

where $b_i \in \{1, \ldots, 64\}$ is the breakpoint value for block i (run-length codes from b_i and up will be eliminated), N is the number of blocks considered. $E^i(k)$ is the value of the DCT coefficient of the kth run in the ith block, and $R_i(b_i)$ denotes the rate required for coding block i using a breakpoint value of b_i.

This constrained minimization problem can be converted to an unconstrained one using Lagrange multipliers; instead of minimizing $\Sigma_i D_i(b_i)$ given $\Sigma_i R_i(b_i)$, we minimize

$$\min \left\{\sum_{i=1}^{N} D_i(b_i) + \lambda \sum_{i=1}^{N} R_i(b_i)\right. \quad (4)$$

Note that the two problems are not equivalent; for some value of λ, however, which our algorithm will have to find, their solutions become identical [25].

The unconstrained minimization problem can be solved using an iterative bisection algorithm (on λ), which at each step k separately minimizes $D_i(b_i) + \lambda R_i(b_i)$ for each block. A similar algorithmic approach, but in a different context, has been used in [9, 24, 25]. A short description of the complete algorithm is as follows. We denote by $R_i^*(\lambda)$ and $D_i^*(\lambda)$ the optimal rate and distortion respectively for block i for that particular λ (i.e., they minimize $D_i + \lambda R_i$). We also denote by $b_i^*(\lambda)$ the breakpoint value that achieves this optimum.

LAGRANGIAN OPTIMIZATION ALGORITHM.

Step 1. Initialization

Set $\lambda_l = 0$ and $\lambda_u = \infty$. If the inequality

$$\sum_{i=1}^{N} R_i^*(\lambda_u) \le R_{\text{budget}} \le \sum_{i=1}^{N} R_i^*(\lambda_l) \quad (5)$$

holds as an equality for either side, an exact solution has been found. If the above does not hold at all, then the problem is infeasible (this can happen if the target rate $\hat{B}$ is too small). Otherwise go to Step 2. Note that these two initial λ's correspond to the minimum and maximum possible breakpoint values (the former minimizes distortion, while the latter minimizes the rate).

Step 2. Bisection and pruning
Compute

$$\lambda_{\text{next}} = \left| \frac{\sum_{i=1}^{N} [D_i^*(\lambda_u) - D_i^*(\lambda_l)]}{\sum_{i=1}^{N} [R_i^*(\lambda_u) - R_i^*(\lambda_l)]} \right| \quad (6)$$

and find $R_i^*(\lambda_{\text{next}})$ and $D_i^*(\lambda_{\text{next}})$ such that $b_i^*(\lambda_u) \leq b_i^*(\lambda_{\text{next}}) \leq b_i^*(\lambda_l)$.
Step 3. Convergence test
If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_u) \text{ or } \sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) = \sum_{i=1}^{N} R_i^*(\lambda_l) \quad (7)$$

then stop; the solution is $b_i^*(\lambda_u)$, $i = 1,$, N. If

$$\sum_{i=1}^{N} R_i^*(\lambda_{\text{next}}) > R_{\text{budget}} \quad (8)$$

then $\lambda_l := \lambda_{\text{next}}$, else $\lambda_u := \lambda_{\text{next}}$.

The bisection algorithm operates on the convex hull of the $R(D)$ curve of each slice. Consequently, points which lie above that and, hence, are not $R(D)$ optimal, are not considered by the algorithm. One can easily verify that actual $R(D)$ curves from real sequences are to a significant degree convex (i.e., only a few points are above the convex hull), particularly for P and B pictures. In some cases, if the $R(D)$ curve of a slice is sufficiently misbehaved, the bisection algorithm can be set off track, with a resulting underutilization of the target bit budget. In order to mitigate this effect and, also, to speed up operation, each iteration considers a continuously shrinking interval of possible breakpoint values ("pruning"). This will result in convergence of the algorithm to a much smaller set of nonconvex points. The computational overhead of the algorithm is small, and convergence is achieved within 8–10 iterations.

The collection of necessary data in (2) requires only parsing of the bitstream up to inverse quantization of the DCT coefficients. Since this represents a small fraction of the complete decoding process, the algorithm has complexity less than that of a decoder. The window N in which the algorithm operates is a design parameter. Since rate shaping is performed on top of encoding (although not necessarily at the same time), it is desirable to minimize the additional delay introduced by the extra processing step. A plausible selection is then a single picture (frame or field). The target bit budget R_{budget} of each picture can be set to $R_{\text{budget}} = (B_T/B)R - R_o$, where R is the size (in bits) of the currently processed picture and R_o is the number of bits spent for coding components of the bitstream that are not subject to rate shaping. R is immediately available after the complete picture has been parsed. Allocated bits that are leftover from one picture are carried over to the subsequent picture.

Since a full resolution picture (704 × 480) may contain up to 15,840 blocks (for a 4:4:4 format), the processing required within each iteration in order to find the breakpoint value that minimizes $D_i(b_i) + \lambda R_i(b_i)$ can be significant. Consequently, it is worth examining *clustering* approaches in which a common breakpoint value is selected for a set of macroblocks. We refer to such algorithms as $C(n)$, where n is the number of sequential macroblocks contained in each cluster. An additional benefit of clustering is that the distortion can be defined on only the luminance part of the signal, hence greatly simplifying the implementation. Clustering, of course, will degrade performance; for example, the C(44) algorithm reduces the quality by about 2 dB, but at a substantial decrease in complexity.

2.3. *Mixed-Mode Rate Shaping*

When all types of picture coding types are used (I, P, and B) the problem is significantly more complex. The decoding process for the original and the rate shaped signal can be described by $P_i = \mathcal{M}_i(P_{i-1}) + e_i$ and $\hat{P}_i = \mathcal{M}_i(\hat{P}_{i-1}) + \hat{e}_i$, where P_i denotes the ith decoded picture (in coding order), $\hat{P}_i$ denotes the rate shaped decoded picture, $\mathcal{M}_i(\cdot)$ denotes the motion compensation operator for picture i, and e_i and $\hat{e}_i$ denote the coded original and rate shaped prediction errors, respectively. The first picture is assumed to be intra-coded, and hence $P_0 = e_0$ and $\hat{P}_0 = \hat{e}_0$. Although, for simplicity a single reference picture is shown above for motion compensation; the expression can be trivially extended to cover the general case (which includes B-pictures).

We can then rewrite (1) as

$$\min_{\sum_{i=1}^{N} R_i(b_i) \leq B_T} \left\| \sum_{p=1}^{M} \mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1}) + \varepsilon_i - \hat{e}_i \right\| \quad (9)$$

where M is the number of pictures over which optimization takes place. Note that in general $\mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1}) \neq \mathcal{M}_i(P_{i-1} - \hat{P}_{i-1})$; i.e., motion compensation is a nonlinear operation, because it involves integer arithmetic with truncation away from zero.

From (9) we observe that, in contrast with the intra-only case, optimization involves the accumulated error $a_i \equiv \mathcal{M}_i(P_{i-1}) - \mathcal{M}_i(\hat{P}_{i-1})$. Furthermore, due to the error accumulation process, rate shaping decisions made for a given picture will have an effect in the quality and partitioning decisions of subsequent pictures. As a result, an optimal algorithm for (9) would have to examine a complete group of pictures (I-to-I), since breakpoint decisions at the initial I-picture may affect even the last B or P picture. Not only

the computational overhead would be extremely high, but the delay would be unacceptable as well.

An attractive alternative algorithm is one that solves (9) on a picture basis, where only the error accumulated from past pictures is taken into account; this algorithm will be referred to as *causally optimal.* Note that in order to accurately compute a_i, two prediction loops have to be maintained (one for a decoder that receives the complete signal, and one for a decoder that receives only partition 0). This is because of the nonlinearity of motion compensation, which involves integer arithmetic with truncation away from zero. With the penalty of some lack in arithmetic accuracy, these two loops can be collapsed together.

The causally optimal problem can be formulated as

$$\min_{\Sigma_{i=1}^{N} R_i(b_i) \le B_T} \left\{ \sum_{i=1}^{N} \hat{D}_i(b_i) \right\} \tag{10}$$

with

$$\hat{D}_i(b_i) \equiv \sum_k A^i(k)^2 + \sum_{k \ge b_i} 2A^i(\mathcal{J}(k))E^i(k) + E^i(k)^2 \tag{11}$$

where N is such that a complete picture is covered, $A^i(k)$ is the kth DCT coefficient (in zig-zag scan order) of the ith block of the accumulated error a_i, and $\mathcal{J}(\cdot)$ maps run/length positions from the prediction error $E^i(\cdot)$ to actual zig-zag scan positions. This minimization problem can be solved using the Lagrangian multiplier approach of Section 2.2, with this new definition for the distortion $\hat{D}$.

An important issue in mixed-mode coding is the target bit budget that will be set for each picture. In a typical situation, I and P picture DCT coding requires a significant number of bits, while B picture sizes are dominated by header and motion vector coding bits. Consequently, B pictures provide much less flexibility for rate shaping. In order to accommodate this behavior, I and P pictures are assigned proportional bit budgets as in Section 2.2; for B pictures the same is done, except when the resulting bit budget is negative, in which case it is set to 0. The negative budget, however, is accounted for, so that the bits spent for the B picture are subtracted from the budget of the immediately following picture. Note that an optimal bit allocation for each picture would be a direct by-product of the optimal (noncausal) algorithm.

The complexity in solving 10 is significant and can be shown to be between that of a decoder and an encoder. In order to examine the benefit of error accumulation tracking, one can apply the intra-only algorithm of Section 2.2 to the mixed-mode case, since the only difference is the accumulated error term a_i. Surprisingly, the results of this *memoryless* mixed-mode partitioning algorithm are almost identical. Figure 2 shows the relevant PSNR values for the "Mobile" sequence; the difference is in general less than 0.1 dB and the curves can hardly be distinguished. It turns out that this holds for a wide range of bit rates (Fig. 3), although the difference increases slightly to 0.2–0.3 dB. This is a very important result, as it implies that we can dispense completely with the error accumulation calculation and its associated computational complexity for a minimal cost in performance; the quality degradation between the causally optimal and memoryless algorithms will be perceptually insignificant across the spectrum of cluster sizes and partition rates.

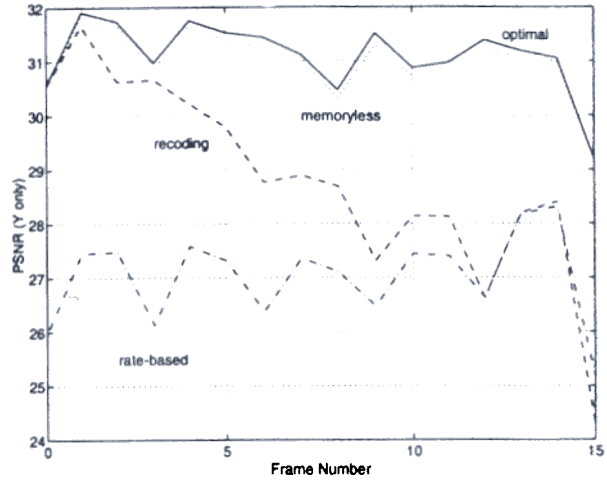

FIG. 2. Results of various rate shaping algorithms on the "Mobile" sequence, MPEG-2 coded at 4 Mbps and rate shaped at 3.2 Mbps.

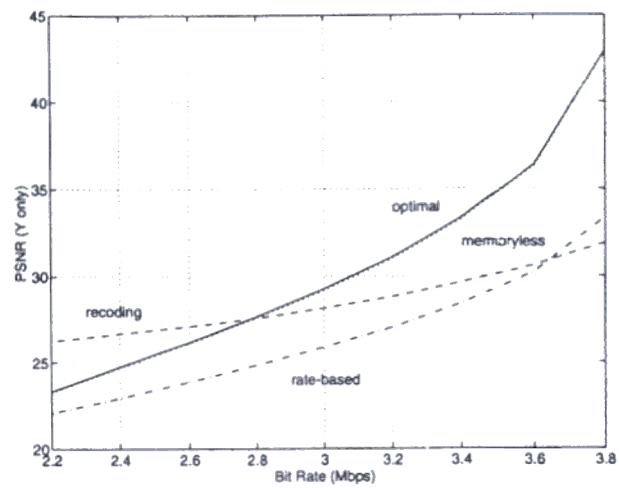

FIG. 3. Results of various rate shaping algorithms on the "Mobile" sequence, MPEG-2 coded at 4 Mbps and rate shaped at various different target bitrates.

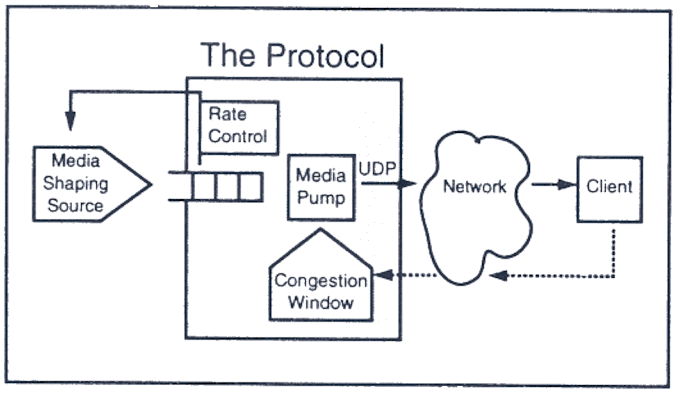

FIG. 4. The Internet-friendly protocol framework.

For comparison purposes, we also examine the performance of a purely rate-based optimization algorithm. Breakpoint selection here is performed proportionally to the number of bits used to originally code each block. Figure 2 depicts the results obtained on the "Mobile" sequence, coded at 4 Mbps and rate shaped at 3.2 Mbps, while Fig. 3 shows average PSNR values for a wide spectrum of rates. Fixed input and output rates have been selected here for simplicity; similar results can be obtained for more complex rate characteristics. All algorithms (except from recoding) are based on $C(1)$ clustering; i.e., breakpoint selection is performed on a macroblock basis. It is important to note that regular recoding gives inferior results to both the optimal and memoryless algorithms for a wide range of rates, while the latter two can hardly be distinguished.

3. THE PROTOCOL

The protocol as shown in Fig. 4 receives data from a media-shaping source, in this case, dynamic rate shaping. This source is responsible for matching the bit rate of the stream to the estimated available bandwidth in the network; the protocol provides this estimate. The source is continually filling the buffer with the media. Meanwhile, the media pump reads the data from the buffer and sends it into the network using UDP/IP. The congestion window is the third part of the server. The media pump only sends out data when the congestion window indicates that more data can enter the network. The congestion window uses feedback from the client, such as acknowledgments of packets received and explicit retransmission requests for packets assumed lost.

If the media shaping source is filling the buffer at a rate R, and the congestion window over time is allowing the media pump to send data out only at a rate $S < R$, then the buffer will begin to fill. The media shaping source should then decrease the rate entering the buffer. A rate control algorithm is responsible for translating the buffer occupancy dynamics into an adjustment to the media shaping source rate to ensure that the buffer does not underflow or overflow.

3.1. *Internet Friendly Bandwidth Estimation*

Although there are many different techniques for finding the available bandwidth in the Internet, probably the most important factor is fairness with existing traffic. When a single network maintains multiple transport protocols, each using a different congestion control algorithm (or none at all); it tends to lead to unfairness [21, 28]. This is because different transport protocols use different definitions of congestion and react differently on detection of congestion. For this reason, it is important that the bandwidth estimation technique does not result in a degradation in the quality of other data transfers on the network.

As real-time services have become more prevalent over the last few years, there has been a growing concern that the current Internet infrastructure may not be able to support them, leading perhaps, to a "congestion collapse." This is a valid concern in general since many real-time services send their bits through the network without concern for congestion control or avoidance.

For this reason, we have designed a technique which is optimally Internet-friendly, in the sense that it shares the available end-to-end bandwidth equally with any other data transfer using TCP, i.e., FTP, HTTP. For example, if between any two network endpoints there is a TCP-based connection and another connection using our protocol, the average bandwidth that both connections will use will be equal. This eliminates concerns about "congestion collapse" since the protocol detects and reacts to congestion in exactly the same way as all other TCP-based non-real-time Internet traffic.

To determine the available bandwidth in the network at a given time we use the TCP congestion window [22], coupled with a rate control algorithm as shown in Fig. 4. The congestion window indicates the number of allowable outstanding unacknowledged packets. This is an indirect indication of the available bandwidth in the network. If the number of outstanding packets is greater than or equal to the congestion window size, then the media pump can send no further data into the network until the situation changes. The congestion window is what paces the delivery of data from the media pump into the network.

Pacing according to a TCP congestion window is exactly what makes the system Internet-friendly. TCP streams work well together and today's Internet is proof of that. They operate according to a greedy but "socially-minded" and cooperative algorithm which attempts to get as much bandwidth as possible, but backs off substantially during congestion. Forcing all data to go through the TCP conges-

tion window can make real-time traffic look as harmless as a file transfer to the network and still maintain relatively low delay [17, 18].

The TCP congestion window is continually changing. It increases quickly at first during the poorly named *slow start phase* and then continues to increase more slowly during the *congestion avoidance phase.* When a packet is lost in the network, TCP assumes this is due to congestion. A packet is lost when the server does not receive an acknowledgment from the client within a specified timeout period. The timeout period is related to the round trip time, the time it takes to send a packet and receive an acknowledgment. The reader is referred to the RFC 793 [22] for more details on the TCP algorithms.

The maintenance of the congestion window and the associated variables are usually kept within the operating system. To implement our protocol we have rebuilt the congestion window and its necessary components in user space. It is important to note that we have not moved all of TCP into user space, but only the algorithms needed to determine the congestion window. For example, we have not included the mandatory retransmissions that TCP normally implements, since we do not want a completely reliable protocol.

Recently, work has been done indicating that Internet-friendly bandwidth estimation can be achieved without using congestion windows, but by using a rate-based algorithm based on packet loss rates [20]. However, simplifying assumptions about the dynamics of TCP are made and the algorithm functions well only during low loss rates. At high loss rates, it overestimates the available bandwidth, defeating the goal of being Internet-friendly. Our protocol is shown in Section 4 to share available bandwidth equally with other TCP connections, which corresponds with our stated goal of being Internet-friendly.

3.2. *Rate Control*

As mentioned before, the rate control algorithm is an essential part of bandwidth estimation. During periods of congestion, the media pump cannot send much data into the network due to the TCP congestion window. When this happens, the buffer in Fig. 4 will start to fill. In this case, a rate control algorithm should force the media shaping source to decrease the rate entering the buffer. If the buffer subsequently begins to empty, the rate control should request an increase in the rate entering the buffer from the source. Clearly, the dynamics of the buffer modulate the rate. Therefore, the estimate of the available bandwidth comes from the rate control algorithm but is an indirect result of the dynamics of the TCP congestion window.

Other work has been done in the design of rate control algorithms. It has been studied extensively in the context of entire networks, where the buffer occupancy from a bottleneck router is used to adjust the rate at which data is injected into the network [1, 2, 4]. Here, the occupancy information traverses the network and is delayed, adding complexity to the algorithms. In our case, the buffer occupancy information is available immediately and simpler models can therefore be used.

Simpler controller models are presented in [2, 16]. In [16], the authors propose the use of a slightly modified proportional control system, where the rate change is based on the distance the current buffer occupancy is away from a desired occupancy. It is called proportional because the change in the rate is proportional to this distance. In [2], the authors show that the pure proportional controller is less than ideal, yielding nondecaying oscillations in the buffer occupancy and rate over time.

A more complex proportional plus derivative (PD) controller can be used which will not exhibit such oscillations [27]. Although these controllers are more effective in time-invariant systems, we have found them to function quite well in the time-varying environment, based on the selection of key tuning parameters. Performance of this algorithm is detailed in Section 4.

The buffer occupancy sampling period for the rate control algorithm depends on the medium. Since MPEG video consists of several different frame types whose sizes vary greatly, estimates of the buffer occupancy must be taken as averages over no less than a one second interval to avoid momentary fluctuations in the buffer occupancy. A small sampling period means that the rate control algorithm can adapt more quickly to sudden changes in the network. However, there is a trade-off since a rapid change in quality is subjectively unpleasant to the user. Larger sampling periods require larger buffers to absorb changes, since the algorithm would respond more slowly.

The buffer in Fig. 4 is being emptied, based on the TCP congestion control window. During interval i, the output rate of the buffer, μ_i, is unknown and the input rate which is controlled entirely by the rate control algorithm is λ_i. At the end of interval i, the buffer occupancy, b_i, can be described by

$$b_i - b_{i-1} = \lambda_i - \mu_i \tag{12}$$

A PD control law would yield a change in the input rate as given by

$$\lambda_{i+1} - \lambda_i = \alpha_0(b_i - b_d) - \alpha_1(b_i - b_{i-1}), \tag{13}$$

where b_d is the desired buffer occupancy. The weighting factors α_0 and α_1 indicate the relative importance of the proportional and derivative factors, respectively. Quick convergence is attained by making α_0 large, while minimizing the overshoot can be accomplished by making α_1 large.

In order to minimize the likelihood of buffer overflow, we chose $b_d = b_{max}/4$, where b_{max} is the total buffer size. This encourages the occupancy to stay at about a quarter full. The choice of b_{max} is based on the original rate of the compressed media, R bps. To further minimize the likelihood of buffer overflow, we chose a b_{max} in bits corresponding to 8 s ($b_{max} = 8R$), although smaller values can be chosen, with a correspondingly higher probability of buffer overflow.

The parameter α_0 was chosen to be 0.005. The reason has both analytical and empirical roots. If the buffer is at the maximum and we ignore the derivative term ($b_i - b_{i-1} = 0$), then Eq. (13) becomes

$$-\lambda_i = -0.005(b_{max} - b_{max}/4). \quad (14)$$

Substituting for b_{max}, we find that the change in the rate is $-3R/100$ bps. For example, if $R = 100$ kbps, then the change in the rate would be -3 kbps. This may not seem like much, but if α_0 is too large, there will be too much overshoot.

For α_1 we use 0.1. If in one interval, the buffer increases by 10% of b_{max} and we ignore the proportional term ($b_i - b_d = 0$), then Eq. (13) becomes

$$\lambda_{i+1} - \lambda_i = -0.1(8R/10). \quad (15)$$

This corresponds to a decrease in the rate by $8R/100$. In other words, an increase in the occupancy of 10% yield a decrease in the rate of 8%. For the same 100 kbps example, the decrease in the rate would be 8 kbps. As shown by our choice of parameters, we have placed more emphasis on the derivative term, since our main goal is to converge quickly and without much overshoot.

Although using the congestion window with the rate control algorithm is an indirect way of calculating the available bandwidth, the only drawback is the need for buffering. But even this is not a major deficit of the protocol, as most systems operating in non-QoS environment employ some buffering to absorb fluctuations in the available bandwidth in the network anyway. In the case of MPEG, the total buffering must be at least a few seconds to absorb the variation in frame sizes, as mentioned earlier. However, for other codecs that do not use inter-frame prediction, much smaller buffering can be used. Applications such as video conferencing are then feasible, since 3 s of latency would be highly unacceptable for conferencing.

3.3. *Retransmissions*

Mandatory retransmissions increase delay, as the client must wait for the retransmitted packet. In general, this is unacceptable for real-time applications. It is certainly the case for video, where we would like to trade quality by losing some data and possibly frames for a lower probability of buffer overflow, which would force the video to stop and then restart playback.

We have used UDP coupled with the TCP congestion window in order to build an Internet-friendly, semi-reliable protocol; TCP is not used for transport. It is semi-reliable because we have added selective retransmissions. The server in TCP maintains an estimate of the round trip time, the time it takes to send a packet and receive an acknowledgement for that packet. In our system, the server sends this information to the client in every packet so that the client has a (somewhat delayed) estimate of the round trip time.

The client then knows best whether or not to request a retransmission. If the client detects that a packet has been lost in the network, it can then determine if there is enough time to request a retransmission and receive the packet before it is needed. For example, if the client has 100 ms of data buffered and knows that the round trip time is 50 ms, then it is likely that the retransmitted packet will arrive on time. If there is not enough buffering to cover the round trip time, then the client will accept the loss and not request a retransmission.

Requesting a retransmission in this case would waste precious bandwidth and would likely not improve the user's experience since the packet would likely not arrive in time. Even if the client requests a retransmission, it is not forced to wait for it to arrive. Error concealment techniques for packet loss are beyond the scope of this paper, but an example for video when part of a frame is lost would be to drop the entire frame.

A retransmission request affects the congestion window similarly to the Fast Recovery Algorithm [26]. In TCP, when a server realizes that it needs to retransmit a packet but that other packets are still arriving at the receiver, the server implements a less severe congestion avoidance algorithm. The idea is that since packets are still arriving at the receiver the network is only moderately loaded and the congestion back-off should reflect this. For this reason, TCP enters the *congestion avoidance phase* rather than the *slow start phase.*

Our protocol does the same thing. When a retransmission request is received and the server is still receiving acknowledgments, the protocol enters the congestion avoidance phase, since this indicates that the network is moderately loaded.

The protocol packet header consists of a 2-byte sequence number, a 4-byte presentation timestamp, and a 4-byte round trip time, measured in milliseconds. Since the underlying protocol is UDP, there is no need for a size field. The sequence number is for packet reordering at the receiver. The timestamp is media dependent, but it is an indication of the presentation time of that packet. The actual size of the payload is also media dependent.

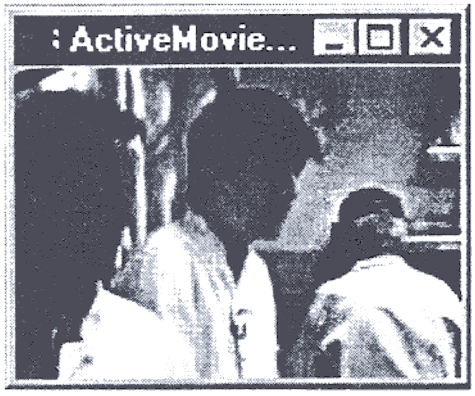

FIG. 5. Screen dump from MPEG video streaming application.

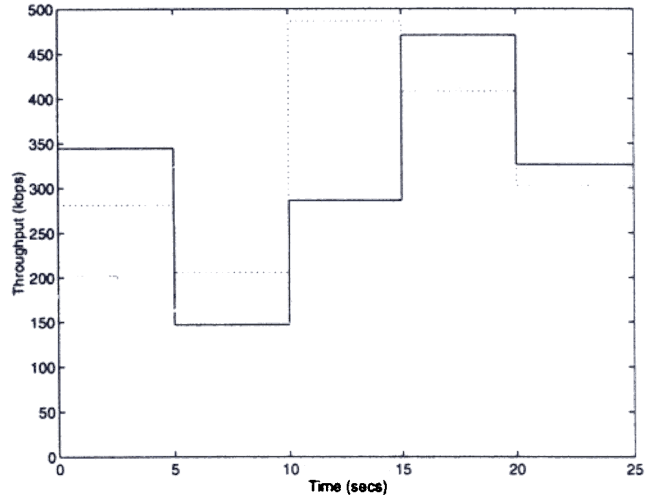

FIG. 6. Two TCP streams sent from the same server to the same client across 19 hops in the Internet, showing that they do in fact share the available bandwidth.

For example, in MPEG video the payload is a frame of MPEG data. If a frame is too large to fit in the payload, it will be fragmented over more than one packet. This creates a stream of packets that have variable lengths. Since TCP normally operates on fixed length packets, alterations were needed. In the case of variable length packets, the congestion window is the number of allowable outstanding unacknowledged bytes. The initial size of the congestion window is defined as 512 bytes, the typical size of a TCP segment. The maximum payload size is also 512 bytes to maintain fairness with TCP.

4. PERFORMANCE RESULTS

We have built an application on top of this protocol for streaming MPEG-1 or MPEG-2 video across the Internet. This application is quite similar to Fig. 4, except that the media shaping source of a dynamic rate-shaping source. Figure 5 is a screen shot of the client window, receiving a streamed MPEG video.

To evaluate our system, we performed three sets of experiments using this application. The first set was to determine if our system was fair in its use of bandwidth, which is the main stated contribution of this paper. The next set of experiments consisted of a client and server with a controllable bottleneck in between, simulating a bottleneck router. The goal here was to determine how quickly the rate control converged to a desired rate and how well it maintained that rate. Finally, since we have a real system, we examined the behavior in the wide area Internet (Fig. 6).

4.1. *Bandwidth Fairness*

The definition of fairness in this environment is that our protocol can share equally the available bandwidth end-to-end with a TCP connection. We used the actual throughput rather than goodput to compare the protocols, since this is where fairness matters most. Also, since TCP requires mandatory retransmissions and our protocol retransmits selectively, our protocol would have a higher output. This is because our protocol uses less throughput on retransmissions than TCP.

We used `tcpdump` to capture how and when the data was sent out. We then averaged the throughput over an interval to calculate the average throughput for each interval. Each test was performed several times. The loss rates were in the range of 0.8% to 2.3%.

We performed four tests here. The results are summarized in Table 1. The goal is to compare different stream types and evaluate how well they share the available bandwidth. As expected the two TCP streams share the available bandwidth quite well. When operating with a pure UDP stream, TCP gets only the bandwidth that the UDP stream does not take. In contrast, our protocol shares the available bandwidth evenly with TCP and itself.

4.2. *Controlled Bottleneck*

Now that we have shown that our protocol competes fairly with TCP and with itself, the next experiments dem-

TABLE 1
Summarized Results for Bandwidth Fairness

Stream 1 type	Stream 2 type	Throughput stream 1	Throughput stream 2	Difference
TCP	TCP	292 kbps	342 kbps	17%
TCP	Pure UDP	205 kbps	810 kbps	295%
TCP	Our Protocol	112 kbps	129 kbps	15%
Our Protocol	Our Protocol	243 kbps	258 kbps	6%

Note. Difference is defined to be the difference between the two throughputs divided by the reference throughput, Stream 1.

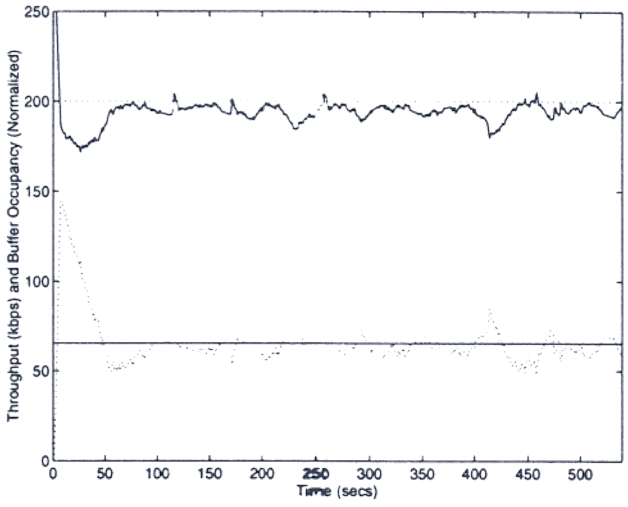

FIG. 7. The rate (solid) and buffer occupancy (dotted) for a stream originally encoded at 250 kbps going through a 200 kbps bottleneck. The buffer occupancy has been normalized to fit on the same graph as the rate. The horizontal lines show the desired rate and buffer occupancy.

onstrate the performance of the rate control algorithm. They were performed with a controlled bottleneck between the client and server, used to simulate a bottleneck router. The bottleneck was set up with a certain bottleneck rate and a maximum buffer size. It reads in packets destined for the client and adds them to the queue. At the same time, it sends out packets onto the network as if the network were operating at the bottleneck rate. It does this by delaying subsequent packets until the time that it would take to send a packet at the bottleneck rate. If the incoming rate is faster than the bottleneck rate, the queue will start to build. If the buffer size is then exceeded, packets are dropped.

Figure 7 shows the effects of an MPEG stream originally encoded at 250 kbps going through a 200 kbps bottleneck with a buffer depth of 10 packets.[3] The duration of the connection is 9 min. The horizontal lines indicate the desired rate and occupancy. The rate moves to a slightly lower value (180 kbps) within the first 10 s, reflecting the bottleneck and the corresponding increase in the buffer occupancy. This is the overshoot period. After the overshoot, the rate settles on the bottleneck rate and stays quite close to it for the remaining 8 min, fluctuating no more than 15 kbps below. The buffer, too, stays within a very narrow range around the desired occupancy. This graph, which is representative of many such tests we performed, demonstrates that the rate control does converge and varies very little around the converged rate. This is not surprising because the PD rate control law is convergent with minimal oscillations in the time invariant case (constant bottleneck rate).

4.3. *Wide Area Network*

In the previous section we showed the convergence of the rate control algorithm, but only under time-invariant conditions. Rather than try to simulate the dynamics of the Internet with our controlled bottleneck, the next experiment is performed using the Internet itself.

This experiment is shown in Fig. 8. The stream was sent from a computer in our laboratory in New York to a computer located 19 hops away in New Jersey, where the packets were read in and sent back immediately to a client in our laboratory again, traversing another 19 hops on the way back. Again the original stream is 250 kbps MPEG.

The experiment was run over a 20-min period, of which this is a 5-minute excerpt. The rate and the buffer occupancy vary substantially, as one would expect given the rate fluctuations in the Internet.

Although the rest of the 20-min period (not shown here) had some buffer overflows, Fig. 8 shows none for the given 5-min period. The goal is to show the advantage of adapting to network conditions. If the original stream had been sent without decreasing its rate, based on the available bandwidth, there would be many more instances of buffer overflow. Figure 9 shows the buffer occupancy from Fig. 8 and the buffer occupancy of a stream which could not adapt to the available bandwidth, sending at the originally encoded rate of 250 kbps. The horizontal line indicates the

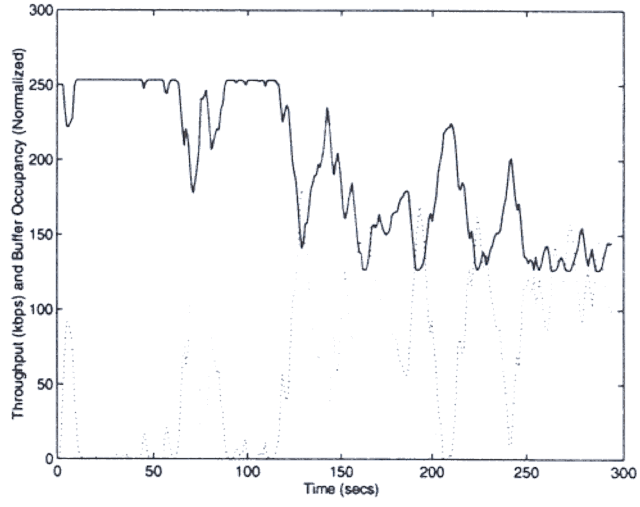

FIG. 8. The rate (solid) and buffer occupancy (dashed) for a stream originally encoded at 250 kbps going through the Internet, traversing 38 hops. The buffer occupancy has been normalized to fit on the same graph as the rate.

[3] The original and rate shaped streams can be retrieved from ftp://wakko.ctr.columbia.edu/share/. The files are fenixc.mpeg and fenixc.200.mpeg, respectively.

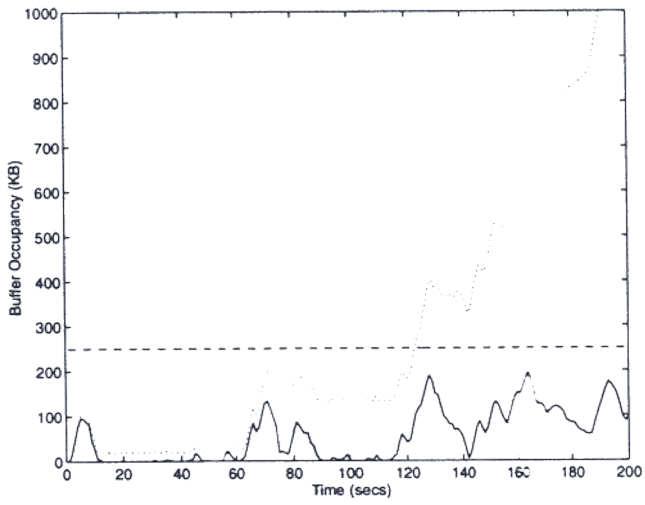

FIG. 9. The buffer occupancy from the previous graph using our protocol (solid) and the buffer occupancy of a stream which did not adapt to the available bandwidth (dotted). Buffer overflow indicates an interruption in playback. The buffer size is shown as a horizontal line.

size of the buffer at the server. When the curve crosses it, the user experiences an interruption in playback. The length of the interruption is the time it takes to refill the buffering at the client.

From 120 s on, the user would have experienced a series of buffer overflows at the server, which would translate into an empty buffer at the client. This is because, although there was enough bandwidth in the first 120 s, the average available bandwidth over the remainder appears to be only 180 kbps. By continuing to adapt to the network, we significantly reduced the number of interruptions that the user would have experienced with nonadaptable video. The protocol does not provide a guaranteed service, but it does provide a service which is much less prone to interruptions in playback. Also any sustained file transfer or web surfing between the same computers would receive the same bandwidth as the video connection.

5. CONCLUDING REMARKS

We have presented a novel application for streaming real time video on the Internet. It consists of an innovative technique for adapting precompressed MPEG video in real time and a protocol which adapts to the available bandwidth in an Internet-friendly way. Dynamic rate shaping was shown as an effective way to create a scalable codec from one which was inherently nonscalable. In addition, a mixed-mode algorithm was presented which, although not optimal, performs almost identically with very low complexity.

The protocol presented was shown to be as friendly to the Internet as TCP, while still managing to be semi-reliable, providing relatively low delay, and significantly decreasing the likelihood of interruptions in playback. It uses the TCP congestion window, coupled with a rate-control algorithm to obtain an estimate of the available bandwidth in the network. It also uses an intelligent selective retransmission scheme, so as not to waste precious bandwidth.

Finally, we performed a variety of tests to experimentally evaluate our protocol design goals of fairness, quick convergence, and stability. We showed analytically and experimentally that these goals were met in both a controlled environment and also in tests in the wide area Internet.

REFERENCES

1. E. Altman *et al.*, Discrete-time analysis of adaptive rate control mechanisms, in *Fifth International Conference on Data Communication Systems and their Performance, October 26–28, 1993,* Vol. C-21, pp. 121–140.
2. L. Benmohamed and S. Meerkov, Feedback control of congestion in packet switching networks: The case of the single congested node, *IEEE/ACM Trans. Networking* **1**(6), 1993, 693–708.
3. R. Bollow, *Video Transmission Using the Available Bit Rate Service,* Master's thesis, Berlin University of Technology.
4. J. Bolot and U. Shankar, Dynamical behavior of rate-based flow control mechanisms, *ACM Comput. Commun. Rev.* **20,** 1990, 35–49.
5. J. Bolot and T. Turletti, A rate control mechanism for packet video in the Internet, in *IEEE Infocom, Toronto, Canada, June 1994.*
6. I. Busse, B. Deffner, and H. Schulzrinne, Dynamic QoS control of multimedia applications based on RTP, in *First International Workshop on High Speed Networks and Open Distributed Platforms, St. Petersburg, Russia, June 1995.*
7. Z. Chen, S. M. Tan, R. H. Campbell, and Y. Li, Real time video and audio in the world wide web, *World Wide Web J.* **1,** 1996.
8. A. Eleftheriadis, Dynamic rate shaping of compressed digital video, Ph.D. thesis, Columbia University, New York, New York, 1995.
9. A. Eleftheriadis and D. Anastassiou, Optimal data partitioning of MPEG-2 coded video, in *Proceedings, 1st IEEE International Conference on Image Processing, Austin, Texas, November 1994,* p. I.273–I.277.
10. A. Eleftheriadis and D. Anastassiou, Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video, in *Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH, April 1995,* pp. 95–106.
11. A. Eleftheriadis and D. Anastassiou, Constrained and general dynamic rate shaping of compressed digital video, in *Proceedings, 2nd IEEE International Conference on Image Processing, Washington, DC, October 1995.*
12. A. Eleftheriadis, S. Pejhan, and D. Anastassiou, Architecture and algorithms of the Xphone multimedia communication system, *ACM/Springer-Verlag Multimedia Systems J.* **2**(2), 1994, 89–100.
13. B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2,* Chapman & Hall, New York, 1997.
14. Information Technology. Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s, ISO/IEC International Standard 11172 (MPEG-1), 1993.

15. Information Technology. Generic coding of moving pictures and associated audio, ITU-T Recommendation H.262, ISO/IEC International Standard 13818 (MPEG-2), 1996.
16. H. Kanakia, P. Mishra, and A. Reibman, An adaptive congestion control scheme for real-time packet video transport, *IEEE/ACM Trans. Networking* **3**(6), 1993, 671–682.
17. S. Jacobs and A. Eleftheriadis, Providing video services over networks without quality of service guarantees, in *World Wide Web Consortium Workshop on Real-Time Multimedia and the Web, Sophia-Antipolis, France, October 24–25, 1996.*
18. S. Jacobs and A. Eleftheriadis, A real time protocol that guarantees fairness with TCP, *IEEE/ACM Trans. Networking,* in press.
19. D. LeGall, MPEG: A video compression standard for multimedia applications, *Comm. ACM* **34**(4), 1991, 46–58.
20. J. Mahdavi and S. Floyd, *TCP-Friendly Unicast Rate-Based Flow Control,* Technical note, end2end-interest mailing list, January 8, 1997.
21. M. Marsan, *et al.,* Simulation analysis of TCP and XTP file transfers in ATM networks, in *Proceedings of Protocols for High Speed Networks, Sophia-Antipolis, France, Oct. 28–30, 1996,* pp. 29–47.
22. J. Postel (Ed.), Transmission control protocol, Inform. Sci. Inst., Request for Comments 793, September 1981.
23. K. K. Ramakrishnan and R. Jain, A binary feedback scheme for congestion avoidance in computer networks, ACM Trans. Comput. Systems **8,** 1990, 158–181.
24. K. Ramchandran and M. Vetterli, Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility, in *Proceedings, Picture Coding Symposium '93, March 1993.*
25. Y. Shoham and A. Gersho, Efficient bit allocation for an arbitrary set of quantizers, *IEEE Trans. Acoust. Speech Signal Process.* **36**(9), 1988, 1445–1453.
26. W. Stevens, TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, Request for Comments 2001, January 1997.
27. R. Vaccaro, *Digital Control, A State Space Approach,* McGraw–Hill, New York, 1995.
28. R. Wilder, Fairness issues for mixed TCP/OSI internets, in *Military Communications in a Changing World, MILCOM'91, McLean, VA,* pp. 177–181.

STEPHEN JACOBS received both the B.S. and M.S. in electrical engineering from Columbia University in 1994 and 1995, respectively. He also holds a B.A. in physics from Bard College. He is currently a Ph.D. candidate in the Electrical Engineering Department at Columbia University and a graduate research assistant in the Image and Advanced Television Laboratory. His research interests include adaptive multimedia protocols and applications for networks without quality of service guarantees. In 1996, Jacobs was awarded the Kodak Fellowship. He is a student member of the IEEE.

ALEXANDROS ELEFTHERIADIS was born in Athens, Greece, in 1967 and received his Ph.D. in electrical engineering from Columbia University in 1995. Since 1995 he has been an assistant professor at Columbia, where he is leading a research team working on media representation and communication as well as multimedia programming, authoring, and playback tools (see http://www.cnmtc.columbia.edu for more information). Dr. Eleftheriadis is a member of the ANSI NCITS L3.1 Committee and the ISO-IEC JTC1/SC29/WG11 (MPEG) Group, where he serves as the Editor of the MPEG-4 Systems specification. His awards include an NSF career award.