A Lossless Smoothing Algorithm for Compressed Video

Simon S. Lam, *Fellow, IEEE*, Simon Chow, and David K. Y. Yau

Abstract— **Interframe coding techniques, such as those used in MPEG video, give rise to a sequence of encoded pictures whose sizes (in number of bits) differ by a factor of ten or more. Buffering is needed to reduce fluctuations in the rate at which video packets are sent to a network connection. In this paper, we design and specify a lossless smoothing algorithm, characterized by three parameters: D (delay bound), K (number of pictures with known sizes), and H (lookahead interval). We prove a theorem which guarantees that, if $K \geq 1$, the algorithm finds a solution that satisfies the delay bound. We present the algorithm's performance from a large number of experiments conducted using MPEG video traces. Lastly, we discuss algorithm implementation.**

I. Introduction

RECENT developments in digital video technology have made possible the storage and communication of full-motion video as a type of computer data, which can be integrated with text, graphics, and other data types. Full-motion video is a sequential display of pictures. In uncompressed form, each picture is a two-dimensional array of pixels, each of which is represented by three values (24 bits) specifying both luminance and color information. From such uncompressed video data, a video encoder produces a coded bit stream representing a sequence of encoded pictures.

In MPEG video [3], for example, there are three types of encoded pictures: I (intracoded), P (predicted), and B (bidirectional).[1] Consider the following sequence of encoded pictures, IBBPBBPBBIBBPBB . . ., where the pattern IBBPBBPBB repeats indefinitely. An I picture is intracoded, i.e., it is encoded, and decoded, without using information from another picture. For P and B pictures, interframe coding is used such that pieces of a P picture are obtained from the preceding I or P picture in the sequence, and pieces of a B picture are obtained from the preceding I or P picture and the subsequent I or P picture in the sequence.[2] A consequence of interframe coding is that an I picture is typically much larger than a P picture (in number of bits), which is much larger than a B picture. Generally, the size of an I picture is larger than the size of a B picture by an order of magnitude.

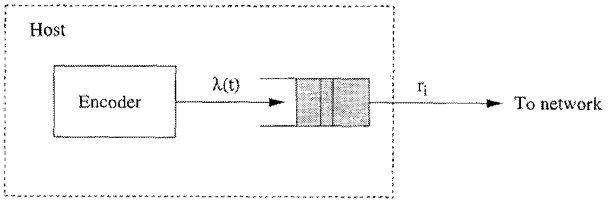


Fig. 1. System model for rate smoothing.

Consider an I picture, which is 200 000 bits long, followed by a B picture, which is 10 000 bits long, in a video sequence. (These are realistic numbers from some of the video sequences we have encoded at a spatial resolution of 640 × 480 pixels; see Section IV.) Suppose the display rate is 30 pictures/s. Sending the I picture in $1/30$ second over a network connection would require a transmission capacity of 6 Mb/s to be allocated to the connection. Then during the next $1/30$ second, the transmission capacity required drops to 0.3 Mb/s for the B picture.

Sending video traffic to a network connection with very large rate fluctuations would adversely affect the network's performance. Consider packet-switching networks that provide a best-effort service. Even though video traffic can conceptually be accommodated without much loss in bandwidth utilization, it is obvious, and has been demonstrated [11], [12], that the statistical multiplexing gain of a finite-buffer packet switch can be improved by smoothing its input flows.[3] Next, consider packet-switching networks that provide per-connection service guarantees. Typically, the service guarantees are provided only if the rate of packet arrivals to a connection conforms to a *flow specification*, or is subject to *usage parameter control* at the user-network interface [1], [10].

In this paper, we present an algorithm for smoothing the rate fluctuations from picture to picture in a video sequence. The algorithm is lossless because smoothing is accomplished by buffering, not by discarding some information. The algorithm can be used for compressed video in general. Its performance, however, is improved by a lookahead strategy which makes use of a repeating pattern of picture types in the sequence (such as the IBBPBBPBB pattern in the above MPEG example). The objective of the algorithm is to transmit each picture in the same pattern at approximately the same rate, while

Manuscript received July 27, 1994; revised December 12, 1995; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Raychaudhuri. This work was supported in part by the National Science Foundation under Grants NCR-9004464 and NCR-9506048, and by a Grant from Lockheed. Early versions of this paper appeared in [6] and [7].

S. S. Lam and D. K. Y. Yau are with the Department of Computer Sciences, the University of Texas at Austin, TX 78712 USA (email: lam@cs.utexas.edu; yau@cs.utexas.edu).

S. Chow is with the Oracle Corporation, Redwood Shores, CA 94065 USA (email: schow@us.oracle.com).

Publisher Item Identifier S 1063-6692(96)07571-1.

[1] We use MPEG video for motivation throughout this paper. The algorithm and theorem presented herein are applicable to compressed video in general.

[2] Each piece consists of 16 × 16 pixels, called a macroblock.

[3] For a specified buffer overflow probability.

```
procedure smooth(H, K: integer; D: real);
var   i, h, sum: integer;
      depart, time, rate, delay,
      lower, upper, lower_old, upper_old: real;
begin i := 0; depart := 0.0; seq_end := false;
   repeat i := i + 1;
      time := max(depart, (i - 1 + K) * tau);
                        {time to begin sending picture i}
      h:=0; sum:=0; lower:=0.0; upper:=∞;
      repeat
         sum := sum + size(i + h, time);
         lower_old := lower; upper_old := upper;
         lower := sum/(D + (i - 1 + h) * tau - time);
         if (time ≥ (K + i + h) * tau) then upper:=∞
            else upper := sum/((K + i + h) * tau - time);
         lower := max(lower, lower_old);
         upper := min(upper, upper_old);
         h := h + 1;
      until (lower > upper) or (h ≥ H);

      if (lower > upper) then
         if (lower > lower_old)
            then rate:=upper          {upper=upper_old}
            else rate := lower
                  {lower = lower_old, upper < upper_old}
      else                                      {h = H}
         if (i = 1) then rate:=(lower+upper)/2;
                                    {rate for first picture}
            else            {possible modification here}
               if (rate>upper) then rate:=upper
                  else if (rate < lower)
                     then rate := lower;
      notify(i, rate);
               {notify transmitter the rate for picture i}
      depart := time + pic_size[i]/ rate;
                           {departure time of picture i}
      delay := depart -(i - 1) * tau   {delay of picture i}
   until seq_end
end; {smooth}
```

Fig. 2. Specification of basic algorithm.

ensuring that any buffering delay introduced by the algorithm is bounded for every picture by a parameter, D, which can be specified *a priori*. Thus the algorithm serves as a mechanism for trading a bounded delay for a smaller peak rate.

The lossless smoothing problem has a relatively straightforward solution if all picture sizes in the video sequence are known *a priori* (see [8]), such as stored video. We do not make this assumption. The main contributions reported herein are: i) the design of an algorithm with no knowledge of the sizes of pictures that have not yet been encoded, ii) an experimental demonstration, using a set of MPEG video traces, that the algorithm is effective—namely, the delay bound is satisfied for individual pictures and picture-to-picture rate fluctuations within a pattern are reduced substantially, and iii) algorithm implementation within the kernel of a workstation.[4]

Note that encoded pictures of the same type also vary in size as the scene in a video sequence changes. More bits are needed to encode pictures of complex scenes and scenes with a lot of motion (P and B pictures in particular). Because scenes typically last for many seconds, buffering would not be a good approach for smoothing such rate fluctuations.

If the rate of a video sequence remains high over a long duration (e.g., a complex scene), various encoder parameters can be adaptively controlled to reduce the encoder output rate. These techniques are said to be *lossy* because in reducing the encoder output rate, some information is discarded. These techniques have been applied to achieve network congestion control using information feedback from network to encoder [2], [5], [9].

In this paper, we focus on the problem of picture-to-picture rate fluctuations, rather than the problems of scene-to-scene rate fluctuations and network congestion control. Our algorithm may be used in addition to lossy techniques proposed for these other problems. The balance of this paper is organized as follows. In Section II, we describe the lossless smoothing problem and a straightforward solution when a delay bound is unimportant. In Section III, the algorithm's theoretical basis is stated as a theorem and a corollary. The algorithm is then designed and specified. In Section IV, we first describe the MPEG video sequences used in our experiments. Experimental results are shown to illustrate the performance and demonstrate the effectiveness of the algorithm. In Section V, we describe how the algorithm is implemented in a workstation kernel, using kernel threads [13], for a proposed flow specification [10]. Section VI has some concluding remarks.

II. Lossless Smoothing

Suppose a video sequence is displayed at the rate of $1/\tau$ pictures per second. τ is called the *picture period*. We assume that the encoding (decoding) time of any picture in the video sequence is less than or equal to τ seconds. We use S_i to denote the size of picture i, $i = 1, 2, 3, \cdots$, which is the number of bits encoding picture i.

Consider a sequence of encoded pictures of sizes, S_1, S_2, $S_3, \cdots$ produced by a video encoder.[5] The size sequence has large fluctuations as a result of interframe coding, e.g., I pictures are much larger than P pictures, which are much larger than B pictures. Assume that in the video sequence, there is a fixed pattern of picture types, which repeats indefinitely. The length of the pattern is denoted by N.

The objective of smoothing is to eliminate the rate fluctuations within a pattern. One way to accomplish this is to buffer one or more pictures (at the sending side of a network connection) so that each picture within the same pattern can be transmitted at the same rate.

To illustrate, consider an MPEG video sequence with $N = 9$ and the repeating pattern IBBPBBPBB. Let $S_i, S_{i+1}, \cdots, S_{i+8}$ be the picture sizes of a particular pattern in the sequence. Thus, the objective of smoothing is to send each picture in this pattern at the following rate

$$\frac{S_i + S_{i+1} + \cdots + S_{i+8}}{9\tau}.$$

That is, the large I picture is transmitted at a lower rate while the small B pictures are transmitted at a higher rate. Note that this averaging of rates is carried out on a pattern by pattern basis to smooth out picture-to-picture rate fluctuations. The rate of the coded bit stream still fluctuates from pattern to pattern. Such fluctuations, however, are inherent characteristics of the video sequence (scene complexity and amount of

[4] Alternatively, the algorithm may be implemented as part of a video encoder.

[5] It is possible that the encoder is adaptively controlled. For example, some lossy technique is concurrently employed to ensure that the size of each I picture is less than a specified maximum value.

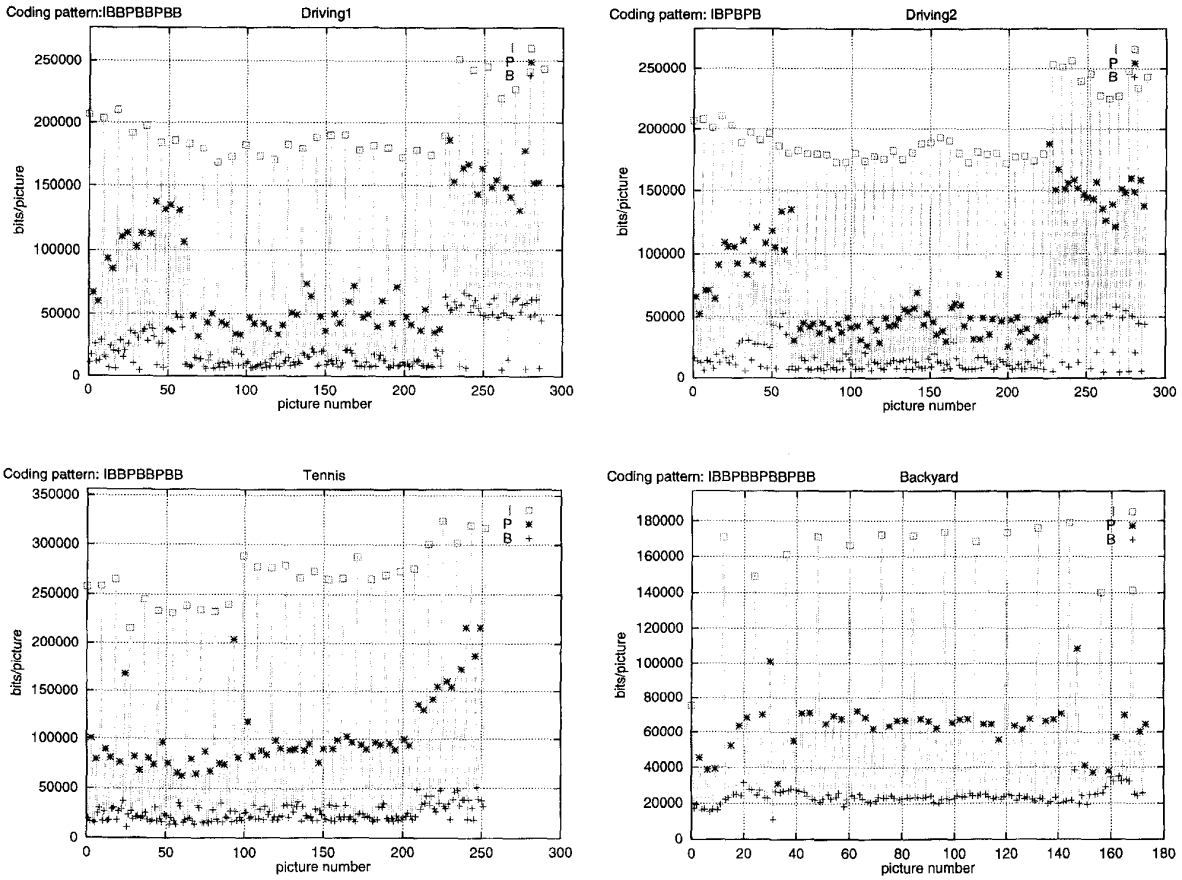


Fig. 3. Four MPEG video sequences.

motion), which cannot be reduced without sacrificing visual quality.

We will refer to the above method as *ideal* smoothing.[6] The ideal method has two disadvantages. First, if the video sequence is generated by a live capture (using a camera), the size of each picture is not known until it has been captured, digitized, and encoded. The pictures in the same pattern would have to be buffered until all have been encoded—and the rate calculated for the pattern—before the first picture in the pattern can be transmitted. In this case, the buffering delay would be very large, and unacceptable for interactive video. Second, the ideal method described above does not pay attention to buffering delays of individual pictures. In particular, it is not possible for a video application to specify an upper bound on such delays.

In the next section, we design an algorithm for smoothing compressed video with the objective that the delay incurred by each picture in the video sequence is less than D, a parameter that can be specified *a priori*.

III. The Algorithm

Both the system model and the algorithm described in this section can be used for compressed video in general. The presence of a repeating pattern in an MPEG video sequence is used to estimate the sizes of pictures that have not been encoded; the MPEG assumption is not needed in the system model, nor in the algorithm.

A. System Model

The model for rate smoothing is a FIFO queue (with some modifications). Input to the queue is from the output of an encoder (see Fig. 1). At time t, let $\lambda(t)$ denote the output rate of the encoder (same as input rate of the queue) in bits/s. We do not know $\lambda(t)$ as a time function. It suffices to assume that the S_i bits encoding picture i arrive to the queue during the time interval from $(i-1)\tau$ to $i\tau$.

The server of the queue represents a channel (physical or logical) which sends the bits of picture i to a network at the rate of r_i bits/s. This rate is calculated for picture i by an algorithm (to be designed and specified) whenever the server can begin sending picture i.

The algorithm has three parameters that can be specified:

K required number of complete pictures buffered in queue before the server can begin sending the next picture ($0 \le K \le N$); specifically, the server can begin sending picture i only if pictures i through $i+K-1$ have arrived (each has been completely encoded);

[6] This method is similar to the CBR video rate controller [4].

D maximum delay specified for every picture in video sequence (seconds);
H lookahead interval, in number of pictures, used by algorithm.

Note that if K is specified to be N, the algorithm has knowledge of all picture sizes needed for ideal smoothing.[7]

The *delay of a picture* is defined to be the time of arrival of its first bit to the queue to the time of departure of its last bit from the queue. (The delay, so defined, includes the picture's encoding delay, queueing delay, and sending delay.) Note that the delay bound D must be specified such that

$$D \geq (K+1)\tau \tag{1}$$

in order for the bound to be satisfiable.

The case of $K = 0$ means that the server can begin sending the bits of picture i buffered in the queue before the entire picture i has arrived. We allow $K = 0$ to be specified for the algorithm. However, using $K = 0$ in an actual system gives rise to two problems. First, buffer underflow is possible unless the encoder is sufficiently fast. Second, the algorithm can ensure that picture delays are bounded by D only if $K \geq 1$ (actually, if and only if $K \geq 1$; see Theorem 1 in Section III-B).

The parameter, H, is for improving algorithm performance by looking ahead (even though only the pattern is known, but not necessarily picture sizes). Its meaning will become clear in Section III-C.

We next define the following notation:

d_i departure time of picture i (the server has just sent the last bit of picture i);
t_i time when server can begin sending picture i.

Additionally, at time t_i, the algorithm calculates the rate r_i. To simplify notation, and without loss of generality, the calculation is assumed to take zero time. The following equation defines the meaning of parameter K

$$t_i = \max\{d_{i-1}, (i-1+K)\tau\}. \tag{2}$$

That is, the server can begin sending picture i only after picture $i-1$ has departed and, if $K > 1$, pictures $i, i+1, \cdots, i-1+K$, have arrived (i.e., encoded and picture sizes are known).

The departure time of picture i is

$$d_i = t_i + (S_i/r_i) \tag{3}$$

and the delay of picture i is

$$\text{delay}_i = d_i - (i-1)\tau. \tag{4}$$

Note that in an actual system, the encoding of picture $i-1+K$ may be complete at time y, such that $(i-2+K)\tau < y \leq (i-1+K)\tau$. Also the first bit of picture i may arrive at time x, such that $(i-1)\tau < x < i\tau$. We use $(i-1+K)\tau$ in (2) and $(i-1)\tau$ in (4) because $\lambda(t)$ is unknown. If either x or y were known and used instead, the delay of each picture may be smaller than the value calculated using (2)–(4), but the difference would be negligible.

[7] Some modification is needed to ensure that the delay of each picture is less than D.

B. Upper and Lower Bounds on Rate

We present an upper bound and a lower bound on the rate r_i that can be selected by an algorithm for sending picture i at time t_i, for all i. The lower bounds are used to ensure that the delay of each picture is less than or equal to D. We say that the algorithm *satisfies delay bound* D if for $i = 1, 2, \cdots$

$$\text{delay}_i \leq D.$$

The upper bounds on rates are used to ensure that the server works continuously. If rates are too large, then the server may send bits faster than the encoder can produce them, forcing the server to idle, i.e., the server cannot send the next picture because the queue does not have K complete pictures.[8] We say that the algorithm *satisfies continuous service* if for $i = 1, 2, \cdots$

$$t_{i+1} = d_i.$$

It might be argued that the delay bound is a more important property than the continuous service property. However, there is no need to choose, because Theorem 1 below shows that both properties can be satisfied. An assumption of Theorem 1 is that S_i is known at time t_i, which can be guaranteed by specifying K to be greater than or equal to one. If S_i is not known at time t_i (i.e., K is specified to be zero), it is easy to construct examples such that the delay bound cannot be satisfied.

Theorem 1: If S_i is known at t_i, and r_i is selected for $i = 1, 2, \cdots, n$ such that conditions (5) and (6) hold

$$r_i \geq \frac{S_i}{D + (i-1)\tau - t_i} \tag{5}$$

$$r_i \leq \frac{S_i}{(i+K)\tau - t_i} \quad \text{if } t_i < (i+K)\tau \tag{6}$$

then for $i = 1, 2, \cdots, n$, the following hold

$$\text{delay}_i \leq D \tag{7}$$

$$t_{i+1} < i\tau + D \tag{8}$$

$$t_{i+1} = d_i. \tag{9}$$

Theorem 1 is proved by induction on n. The proof is given in the Appendix.

We use r_i^L and r_i^U to denote the lower bound in (5) and the upper bound in (6), respectively. For these upper and lower bounds, we say that a bound is *well defined* if its denominator is positive; see (5) and (6). In Theorem 1, (8) guarantees that the lower bounds are all well defined. As for the upper bounds, many in (6) may not be well defined. These are defined as follows

$$r_i^U = \infty \quad \text{if } t_i \geq (i+K)\tau.$$

Because of (1), the following corollary is immediate.

Corollary 1: For all $i = 1, 2, \cdots, n$,

$$r_i^L \leq r_i^U. \tag{10}$$

Corollary 1 implies that both the delay bound and the continuous service property can be satisfied.

[8] For $K = 0$, buffer underflow may occur.

C. Lookahead to Improve Algorithm Performance

Theorem 1 requires that the rate for picture i be chosen from the interval $[r_i^L, r_i^U]$, which may be large if $D > (K+1)\tau$. This flexibility can be exploited to reduce the number of rate changes over time. Suppose the sizes of pictures i, $i+1$, $i+2, \cdots$ are known. The algorithm can be designed to find a rate for sending pictures i through $i+h$, for as large a value of h as possible.

In our system model, however, the size of picture j, $j > i+K-1$, may not be known at time t_i. Specifically, for $K=1$, it is likely that S_j, $j > i$, has to be estimated. Fortunately, Theorem 1 requires only S_i to be known at t_i. Sizes of pictures arriving in the future may be estimated without affecting Theorem 1.

In what follows, we derive a set of upper bounds and a set of lower bounds from $S_i, S_{i+1}, S_{i+2}, \cdots$, where S_j, $j > i+K-1$, may be an estimate. There are many ways to estimate the size of a picture from past information. In the experiments described in Section IV, the size of picture j, if not known at t_i, was estimated to be S_{j-N}. This is a simple estimate which uses the fact that pictures $j-N$ and j are of the same type (I, B, or P) in MPEG video. They are about the same size unless there is a scene change in the picture sequence from $j-N$ to j.

If all pictures in the future are sent at the rate r_i, the (approximate) delay of picture $i+h, h=0,1,2,\cdots$, is

$$t_i + \frac{\sum_{m=0}^{h} S_{i+m}}{r_i} - (i-1+h)\tau. \tag{11}$$

Requiring the above to be $\leq D$, we have

$$r_i \geq \frac{\sum_{m=0}^{h} S_{i+m}}{D + (i-1+h)\tau - t_i} \tag{12}$$

where the lower bound on r_i will be denoted by $r_i^L(h)$.

The (approximate) departure time of picture $i+h$ is

$$d_{i+h} = t_i + \frac{\sum_{m=0}^{h} S_{i+m}}{r_i}.$$

The continuous service property requires that $d_{i+h} \geq (i+h+K)\tau$, which can be satisfied by requiring

$$r_i \leq \frac{\sum_{m=0}^{h} S_{i+m}}{(i+h+K)\tau - t_i} \quad \text{if } t_i < (i+h+K)\tau \tag{13}$$

where the upper bound on r_i will be denoted by $r_i^U(h)$ if $t_i < (i+h+K)\tau$; else, $r_i^U(h)$ is defined to be ∞.

Note that $r_i^L(0)$ and $r_i^U(0)$ are equal to the lower bound r_i^L and upper bound r_i^U, respectively, given in Theorem 1. Also, only $r_i^L(h)$ and $r_i^U(h)$ for $h=0,1,\cdots,K-1$ are accurate bounds; the others, calculated using estimated picture sizes, are approximate.

A strategy to reduce the number of rate changes over time is to first find the largest integer h^* such that

$$\max_{0 \leq h \leq h^*} r_i^L(h) \leq \min_{0 \leq h \leq h^*} r_i^U(h). \tag{14}$$

The rate r_i for picture i is then selected such that for $h = 0,1,\cdots,h^*$

$$r_i^L(h) \leq r_i \leq r_i^U(h).$$

Note that for $K \geq 1$, the selected rate satisfies

$$r_i^L = r_i^L(0) \leq r_i \leq r_i^U(0) = r_i^U.$$

Therefore, the hypothesis of Theorem 1 holds and the delay bound D as well as the continuous service property are satisfied even though picture sizes (namely, S_j, $j > i$) are estimated.

To minimize delay, we would like to use $K=1$ in the algorithm, in which case most picture sizes are estimated. For this reason, the smoothing algorithm in Section III-D is designed with a parameter H which can be specified. Instead of searching for the largest h^* satisfying (14), the search is limited to a maximum value of $H-1$. For MPEG video, we conjecture that there is no advantage in having H greater the size of a pattern (N) because picture sizes are estimated using past information. We conducted experiments to study this conjecture and found that it is supported by experimental data; the results are presented in Section IV.

D. Algorithm Design and Specification

The smoothing algorithm is designed using (2)–(4), (12)–(14), Theorem 1, and Corollary 1. A specification of the *basic algorithm* is given in Fig. 2. The following are assumed to be global variables:

pic_size: **array** [*index*] **of** *integer*;
seq_end: *boolean*;
tau: *real*;

The value of *pic_size*[i] is S_i in the system model, the value of *tau* is the picture period, and *seq_end*, initially *false*, is set to *true* when the algorithm reaches the last picture of a video sequence.

There are three functions in the specification: *max*, *min*, and *size*. In particular, *size*(j,t) returns, at time t, either the actual size of picture j or an estimated size (in number of bits). For the experimental results presented in Section V, we used the following simple estimation based upon the fact that a fixed pattern of N picture types repeats indefinitely in each video sequence

if$(t \geq j * tau)$ **then return** *pic_size*[j]
else return *pic_size*[$j-N$].

For the initial part of a video sequence, where *pic_size*[$j-N$] is not defined, each I picture is estimated to be 200 000 bits, each P picture 100 000 bits, and each B picture 20 000 bits. These estimates are far from being accurate for some video sequences. But by Theorem 1, they do not need to be accurate.

Lastly, we use *notify*(j,r) to denote a communication primitive which notifies a transmitter that picture j is to be sent at rate r.

Note that the inner **repeat** loop calculates the bounds in (14). The loop has two exit conditions. The exit condition, (*lower* > *upper*), corresponds to h^* in (14) being less than $H-1$; when this happens (called *early exit*), it can be proved that one of these two conditions holds:

- *lower* > *lower_old* and *upper* = *upper_old*
- *lower* = *lower_old* and *upper* < *upper_old*

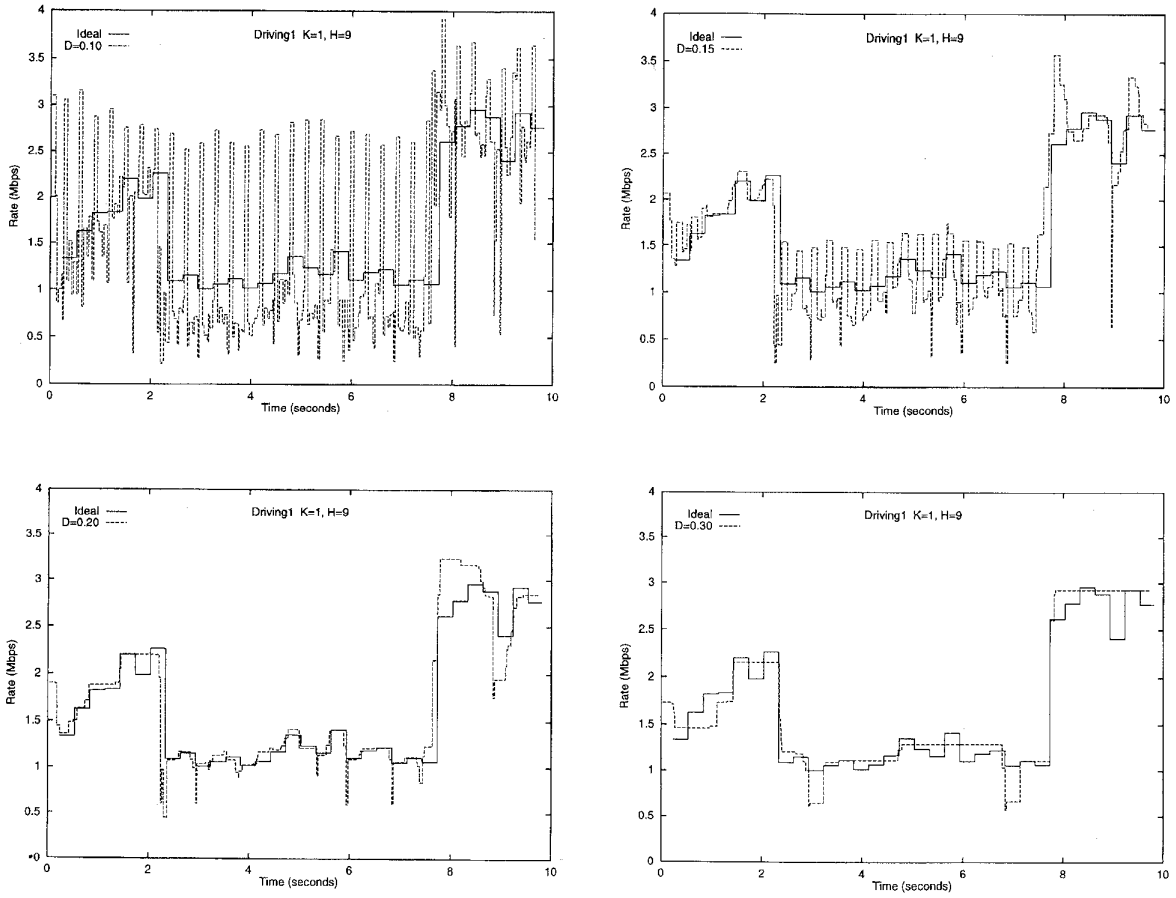


Fig. 4. Rate as a function of time for four delay bounds (Driving1 sequence, basic algorithm).

The selection of r_i in each case is designed to minimize the number of rate changes over time.

The second exit condition corresponds to h^* in (14) being larger than or equal to $H - 1$ (called *normal exit*); in the algorithm, the search for h^* stops at $h = H - 1$ because the lookahead interval is limited to H pictures. Upon normal exit, r_i is selected to be the same as r_{i-1}, i.e., no rate change unless the current value of *rate* is larger than *upper* or smaller than *lower*. This selection strategy is designed to minimize the number of rate changes.

We also investigated a variation of the basic algorithm such that the moving average calculated using

$$rate := sum/(N * tau) \tag{15}$$

is selected for r_i (unless the moving average is larger than *upper* or smaller than *lower*). To modify the algorithm, the assignment statement in (15) replaces the comment "{possible modification here}" in procedure *smooth*. The modified algorithm produces numerous small rate changes over time, but its rate $r(t)$, as a function of time, tracks the rate function of ideal smoothing more closely than the basic algorithm. In particular, the area difference (a performance measure defined in Section IV) is smaller.

IV. Experiments

To show that the smoothing algorithm is effective and satisfies the correctness properties given in Theorem 1, we performed a large number of experiments using four MPEG video sequences. Some of our experimental results are shown and discussed below. For all experiments, the picture rate is 30 pictures/s.

A. MPEG Video Sequences

1) Driving1 ($N = 9$) and Driving2 ($N = 6$): This video was chosen because we thought that it would be a difficult one to smooth. There are two scene changes in the video. Initially, the scene is that of a car moving very fast in the countryside. The scene then changes to a close-up of the driver, and then changes back to the moving car. This video is encoded twice, using different coding patterns, to produce two MPEG video sequences. Note, from Fig. 3, that the scene changes give rise to abrupt changes in picture sizes. In particular, P and B pictures in the driving scenes are much larger than P and B pictures in the close-up scene. The pictures were encoded with a spatial resolution of 640 × 480 pixels.

2) Tennis ($N = 9$): This video shows a tennis instructor initially sitting down and lecturing. He then gets up to move away. There is no scene change in the video. But as the

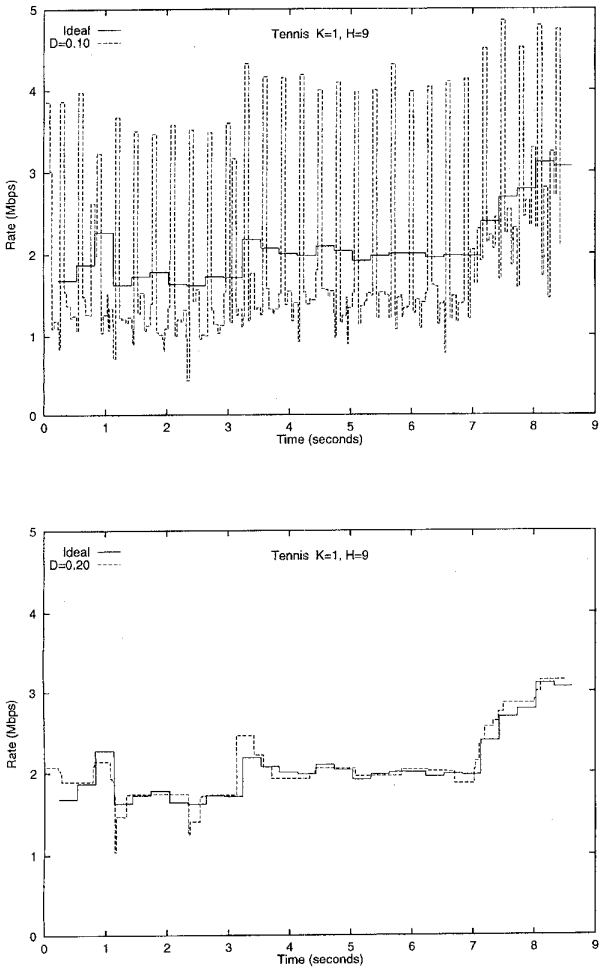


Fig. 5. Rate as a function of time for two delay bounds (Tennis sequence, basic algorithm).

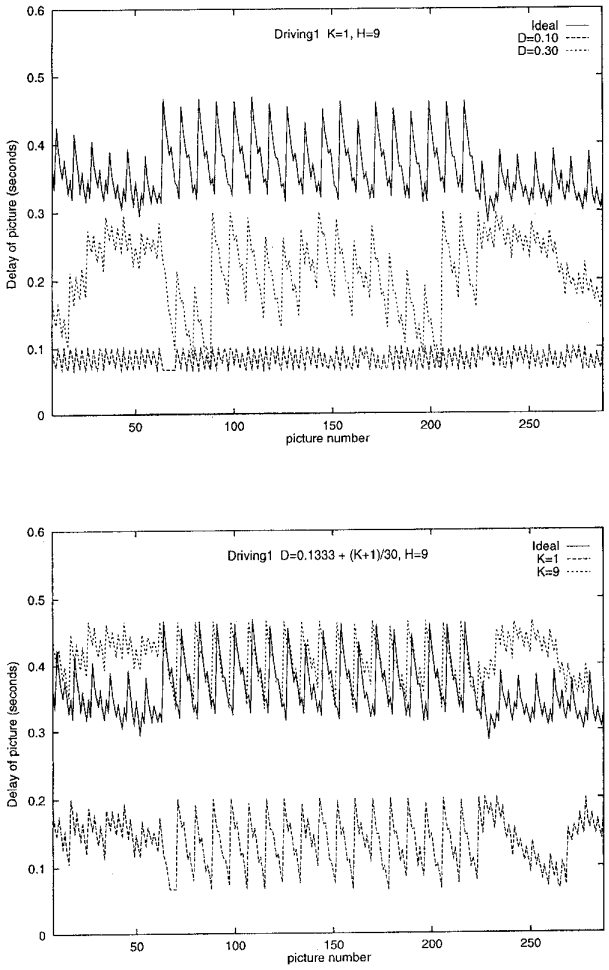


Fig. 6. Delays of pictures in Driving1 sequence (basic algorithm).

instructor gets up, his motion gives rise to increasingly large P and B pictures. These changes in picture sizes are gradual. However, there are two isolated instances of large P pictures in the first half of the sequence. The pictures were encoded with a spatial resolution of 640 × 480 pixels.

3) Backyard ($N = 12$)*:* There are also two changes of scene in this video. Initially, the scene is that of a person in a backyard. The scene changes to two other people in another area of the backyard, and then changes back to the first person. The backgrounds of both scenes are complex with many details. While there are movements, the motion is not rapid. The pictures were encoded with a spatial resolution of 352 × 288 pixels.

B. Performance of the Basic Algorithm

For the Driving1 sequence, Fig. 4 shows bit rate as a function of time for $K = 1$, $H = 9$, and four values of the delay bound D, in seconds. In each case, we compare the rate function from the basic algorithm, denoted by $r(t)$, with the rate function from ideal smoothing, denoted by $R(t)$. From Fig. 4, we see that the "smoothness" of $r(t)$ improves as the delay bound is relaxed. (We will define some quantitative measures of smoothness below.)

For $D = 0.1$, $r(t)$ does not look smooth at all, even though it is a lot smoother than the rate function $\lambda(t)$ of the MPEG encoder output. (Without smoothing, the largest I picture would require over 7.5 Mb/s to send in $1/30$ second.) Note that the improvement in smoothness from $D = 0.2$ to $D = 0.3$ is not significant. Therefore, $D = 0.2$ would be an excellent parameter value to use if a delay of up to 0.2 second (which includes encoding delay) is an acceptable price to pay for a peak rate of about 3.5 Mb/s.

For the Tennis sequence, the results are very similar. Fig. 5 shows $r(t)$ and $R(t)$ for $K = 1$, $H = 9$, and two values of the delay bound D.

Note that the smoothed rate function of the Driving1 sequence varies from a maximum of about 1 Mb/s to 3 Mb/s. These variations are due to differences in the content and motion of scenes. The smoothed rate function of the Tennis sequence varies from a maximum of about 1.5 Mb/s to 3 Mb/s. The peak rate is about the same in the two video sequences because they were encoded with the same spatial resolution (640 × 480) and same quantizer scale (4 for I, 6 for P, and 15 for B).

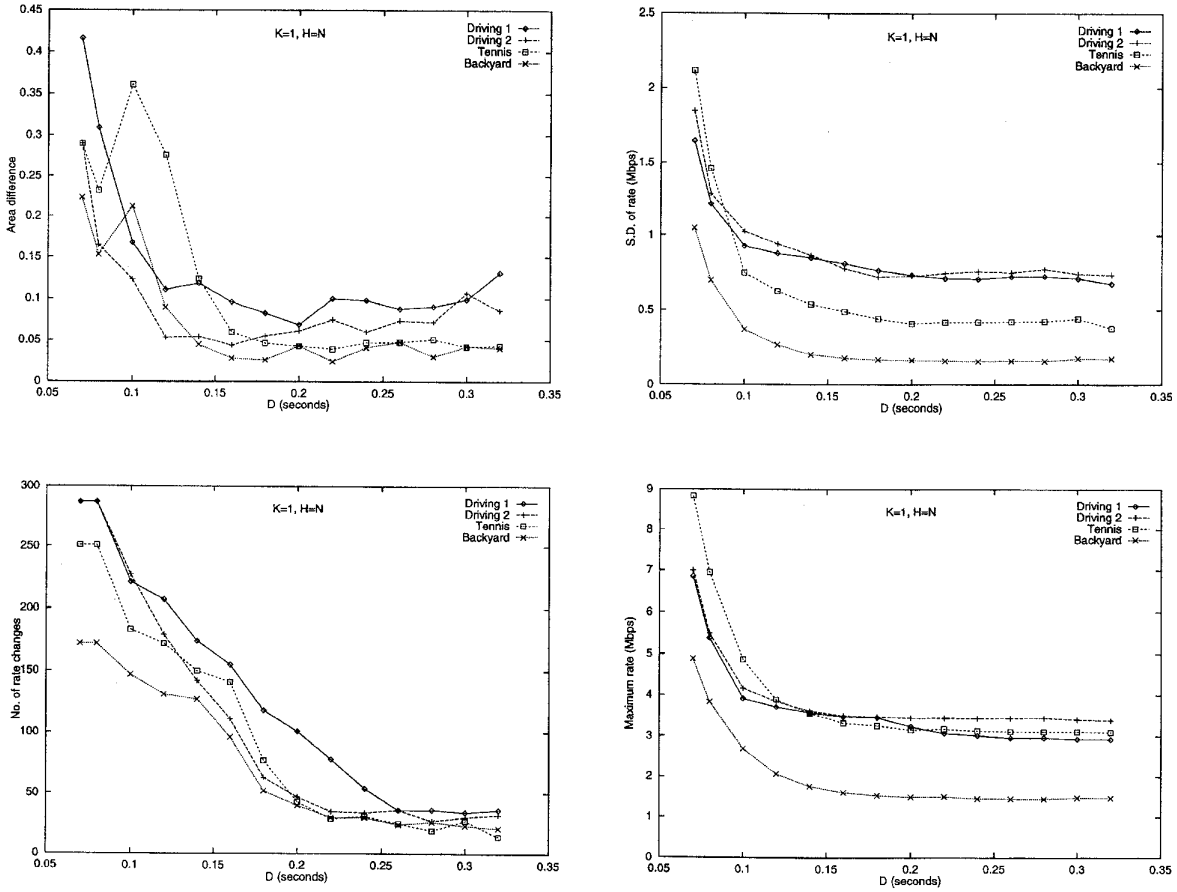


Fig. 7. Performance of the basic algorithm as a function of delay bound D.

For the Driving1 sequence, Fig. 6 shows the delays of pictures for two comparisons. In the upper graph, we compare these three cases:

1) $D = 0.1$, $K = 1$, $H = 9$, basic algorithm.
2) $D = 0.3$, $K = 1$, $H = 9$, basic algorithm, and
3) ideal smoothing.

As shown, the delays of pictures are bounded by 0.1 second and 0.3 second as specified for the basic algorithm. For ideal smoothing, picture delays are large, due to the requirement that pictures in the same pattern are buffered until all have arrived before the first picture in the pattern can be transmitted.

In the lower graph of Fig. 6, we compare these three cases:

1) $K = 1$, $H = 9$, $D = 0.1333 + (K + 1)/30$, basic algorithm,
2) $K = 9$, $H = 9$, $D = 0.1333 + (K + 1)/30$, basic algorithm, and
3) ideal smoothing.

For $K = H = N = 9$, the smoothing algorithm does not estimate picture sizes. In this case, the basic algorithm is very similar to ideal smoothing.[9]

A comparison of the delays for the two cases, $K = 1$ and $K = 9$, shows the desirability of using $K = 1$. The *slack* in the delay bound is chosen to be the same, 0.1333 second, so that the smoothness of $r(t)$ is about the same in both cases (see discussion on Fig. 9).

No "delay bound violation" has been observed in any of our experiments where $K \geq 1$. This is not surprising, since the absence of delay bound violation is guaranteed by Theorem 1 if $K \geq 1$. For $K = 0$, however, we did observe some delay bound violations when the slack in the delay bound was deliberately made very small.

Different quantitative measures can be defined to characterize the effectiveness of smoothing. We use four of them to study algorithm performance as each of the parameters, D, H, K, varies. The first measure is defined as follows:

$$\text{Area difference} = \frac{\int_0^T [r(t) - R(t + (N-K)\tau)]^+ \, dt}{\int_0^T R(t + (N-K)\tau) \, dt} \quad (16)$$

where T denotes the time duration of the video sequence. Note that with ideal smoothing, picture 1 begins transmission $(N - K)\tau$ second later than if the basic algorithm were used. Therefore the rate function from ideal smoothing is shifted by this much time in (16). Only the positive part of the difference between $r(t)$ and $R(t)$ is used in (16) because of the following:

$$\int_0^T [r(t) - R(t + (N-K)\tau)] dt = 0.$$

[9]They are not identical, because ideal smoothing as described in Section II does not try to keep the delay of each picture less than a specified bound D.

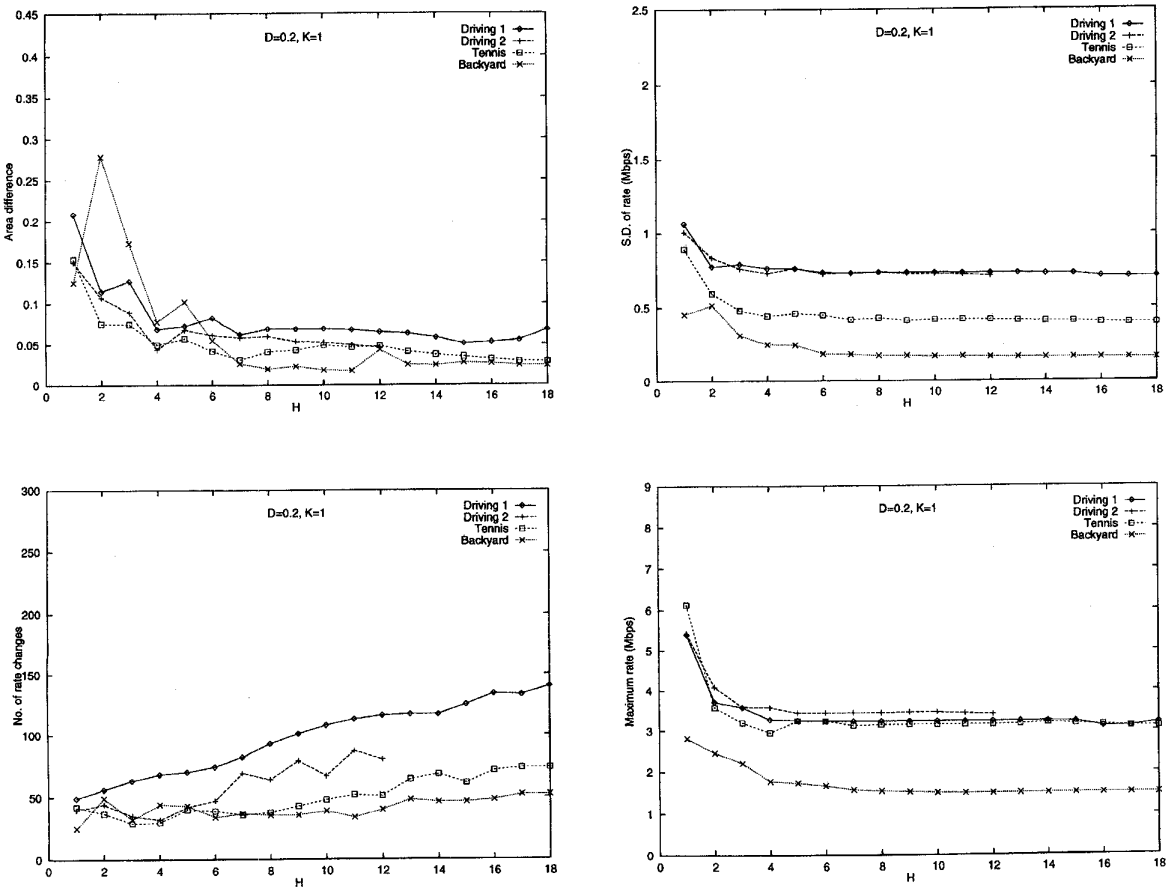


Fig. 8. Performance of the basic algorithm as a function of parameter H.

We use three other measures

1) the number of times $r(t)$ is changed by the algorithm over $[0, T]$,
2) the maximum value of $r(t)$ over $[0, T]$, and
3) the standard deviation (S.D.) of $r(t)$ over $[0, T]$.

Fig. 7 shows the four quantitative measures as a function of delay bound D for the four MPEG video sequences. All four measures indicate that as the delay bound is increased (relaxed), the rate function $r(t)$ becomes more smooth. The Backyard sequence appears to be the easiest to smooth. For the three MPEG video sequences encoded at a spatial resolution of 640 $\times$ 480 pixels, the maximum smoothed rate is about 3 Mb/s. For the Backyard sequence encoded at a spatial resolution of 352 $\times$ 288 pixels, the maximum smoothed rate is about 1.5 Mb/s, which is about the target rate of the MPEG standard. The maximum (smoothed) rate versus D curves in Fig. 7 represent a valuable design tradeoff made possible by our lossless smoothing algorithm.

Fig. 8 shows the quantitative measures as a function of the lookahead interval, H, for the four MPEG video sequences. In Section III-C, we conjectured that because most picture sizes are estimated using past information, there is no advantage in having H larger than the size of the repeating pattern (N). Our experimental data support this conjecture. In Fig. 8, the area difference, standard deviation of rate, and maximum rate do not show any noticeable improvement for values of H larger than N. In fact, the number of rate changes increases as H increases.

K should be as small as possible to reduce picture delay. Theorem 1 requires $K \geq 1$. We conducted experiments to investigate whether there is any improvement in smoothness of $r(t)$ from using $K > 1$. Fig. 9 shows that there is a small improvement as K increases, but barely noticeable. Note that the delay bound is $D = 0.1333 + (K+1)/30$, with a constant slack of 0.1333 for all cases. We conclude that $K = 1$ should be used.

V. Application and Implementation

We first discuss how to apply the lossless smoothing algorithm at a particular user-network interface, namely, the proposed flow specification in [10]. Algorithm implementation in a workstation kernel is then briefly described.

The relevant parameters in the flow specification [10] are the following:[10]

1) maximum transmission rate, $r_{\max}$ (bits/s),
2) token bucket rate, ρ (bits/s), and
3) token bucket size, σ (bits).

[10]The parameters in [10] are specified in *bytes* and *bytes/s*. We use *bits* and *bits/s* herein to be consistent with the parameters in Sections III and IV.

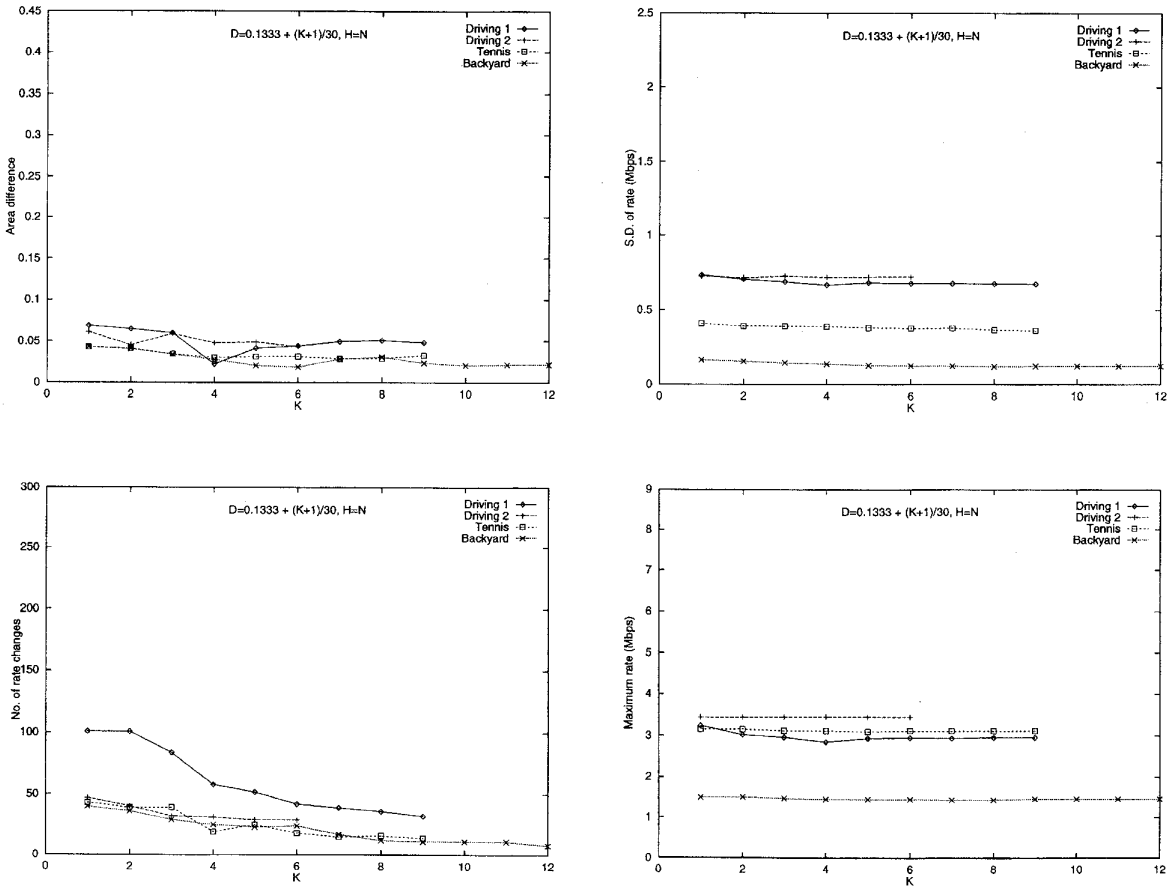


Fig. 9. Performance of the basic algorithm as a function of parameter K.

The maximum transmission rate specifies a bound on how fast successive packets may be sent to a network connection. The token bucket parameters specify the following *leaky bucket constraint*: The amount of data sent to the network connnection, over any arbitrary time interval t, cannot exceed $\sigma + \rho t$.

Consider a video application sending the packets of a video sequence to a network connection. The lossless smoothing algorithm can be used to assist in controlling the rate at which video packets arrive to the network connection such that the flow specification is satisfied. From Fig. 7, note that the maximum rate of a smoothed MPEG sequence decreases as D is increased from 0 to 0.2 second and levels off beyond 0.2 second. Thus the lossless smoothing algorithm can be used to trade an increase in picture delay for a decrease in maximum transmission rate. However, the algorithm alone may not be sufficient to satisfy an arbitrary $r_{\max}$ constraint in a flow specification. The video encoder parameters (spatial resolution, temporal resolution, quantizer scale, etc.) must be chosen a priori such that the $r_{\max}$ constraint is satisfiable, i.e., the maximum rate curve of the smoothed video sequence, such as those in Fig. 7, dips below $r_{\max}$ as D is increased.

To satisfy the leaky bucket constraint for an MPEG video sequence, it is necessary to bound the size (number of bits) of the largest picture in the video sequence such that, for all i

$$S_i \leq \sigma + \rho(D - K\tau). \tag{17}$$

In general, it is necessary that, for any n consecutive pictures in the sequence, $n \geq 1$, for all i

$$(S_i + S_{i+1} + \cdots + S_{i+n-1}) \leq \sigma + \rho(D + (n - K - 1)\tau). \tag{18}$$

Thus for a particular choice of D, the video encoder may need to employ lossy techniques to enforce an upper limit on the size of pictures.

The lossless smoothing algorithm can be implemented as part of a video encoder (as described in [4]). Our current implementation of the algorithm, however, is in the modified kernel of a Sparc 10 workstation (running SunOS 5.3, the operating system component of Solaris 2.3). Our modified kernel was designed and implemented to provide efficient support and rate-based flow control for distributed multimedia [13]. In particular, the user process of a video application is allocated a send queue (implemented using buffers co-mapped to both user and kernel space) which is periodically served by a *kernel thread*. Lossless smoothing is implemented as part of the kernel thread. Since the kernel thread executes periodically (rather than continuously as assumed in Section III), the rate r_i is measured in bytes/period and is enforced (approximately)

once per period when the kernel thread executes. A typical value for the scheduling period is 10 ms in our current implementation.

VI. Conclusion and Related Work

As part of our research project on the design of efficient operating system support and network protocols for distributed multimedia, we studied MPEG video. We found that interframe coding techniques, such as those specified by MPEG, give rise to a coded bit stream in which picture sizes (in number of bits) differ by a factor of ten or more. As a result, some buffering is needed to smooth the picture-to-picture rate fluctuations in the coded bit stream; otherwise, the large rate fluctuations would make it very difficult to allocate a communication channel with appropriate quality-of-service guarantees.

Our algorithm is designed to satisfy a delay bound, D, which is a parameter that can be specified by a video application. The algorithm is characterized by two other parameters, K, the number of pictures with known sizes, and, H, a lookahead interval for improving algorithm performance. We have proved a theorem which states that if $K \geq 1$, then our algorithm satisfies both the delay bound D and a continuous service property.

Although our system model and algorithm, as well as Theorem 1, were motivated by MPEG video, they are applicable to any compressed video. We make use of the assumption that there is a fixed pattern of picture types, which repeats indefinitely in the video sequence, to estimate picture sizes. Such size estimates are used in a lookahead strategy to improve algorithm performance.

The problem of smoothing was analyzed by Ott *et al.* [8], where picture sizes in a video sequence are assumed to be known a priori. The parameter K is absent in their model, and there is no notion of a repeating pattern [8]. From a practical point of view, K is a crucial parameter for any smoothing algorithm. Furthermore, Theorem 1 shows that there is no need to assume all picture sizes to be known a priori. Instead, we use a fixed pattern with estimated picture sizes in our algorithm.

We conducted a large number of experiments using four MPEG video traces to study the performance of our algorithm. We found that the algorithm is effective in smoothing rate fluctuations, and behaves as described by Theorem 1. Experimental data suggest that the following choice of parameters provides a smooth rate function: $K = 1$, $H = N$, and $D = 0.2$. The delay bound includes the encoding delay of each picture. A larger delay bound does not seem to provide any noticeable improvement in the smoothness of the resulting rate function. In conclusion, the lossless smoothing algorithm is an effective mechanism for trading an increase in bounded delay for a decrease in the peak transmission rate.

Lastly, we note that the lossless smoothing algorithm was not designed to smooth rate fluctuations from scene to scene in a video sequence, and is not intended to be the only means of source control to alleviate network congestions. To address these other problems, various lossy techniques [2], [5], [9] will have to be used in addition to lossless smoothing.

Appendix
Proof of Theorem 1

The proof is by induction on n.

Base Case: $n = 1$

1) $t_1 = K\tau$ [$d_0 = 0, i = 1$ in (2)]
2) $r_1 \geq \frac{S_1}{D - t_1} = \frac{S_1}{D - K\tau}$ [assumption, $i = 1$ in (5), step 1]
3) $r_1 \leq \frac{S_1}{(1+K)\tau - t_1} = \frac{S_1}{\tau}$ [assumption, $i = 1$ in (6), step 1]

We need to prove (7), (8), and (9).

4) r_1^L is well defined [step 2 and (1)]
5) $\text{delay}_1 = t_1 + (S_1/r_1)$ [$i = 1$ in (4)]
$\leq t_1 + D - K\tau$ [steps 2 and 4]
$= D$ [step 1]
6) $t_2 = \max\{d_1, (1+K)\tau)\}$ [$i = 2$ in (2)]
7) $d_1 = \text{delay}_1 \leq D$ [$i = 1$ in (4) and step 5]
8) $t_2 < D + \tau$ [steps 6 and 7, (1)]
9) r_1^U is well defined [$\tau > 0$, step 3]
10) $d_1 \geq t_1 + \tau$ [(3), steps 3 and 9]
11) $d_1 \geq (1+K)\tau$ [steps 1 and 10]
12) $t_2 = d_1$ [$i = 2$ in (2), step 11]

Steps 5, 8, and, 12 demonstrate that (7), (8) and (9) hold for $i = 1$. Proof of base case is complete.

Induction Step:

13) Theorem 1 holds for $i = 1, 2, \cdots, m-1$. [$n = m - 1$]

It suffices to prove that if r_m is selected using (5) and (6) for $i = m$, then (7), (8), and (9) hold for $i = m$.

14) r_m^L is well defined [step 13, $i = m - 1$ in (8)]
15) $\text{delay}_m = t_m + (S_m/r_m) - (m-1)\tau$ [(3) and (4)]
$\leq t_m + D + (m-1)\tau - t_m - (m-1)\tau$ [$i = m$ in (5), step 14]
$= D$
16) $t_{m+1} = \max\{d_m, (m+K)\tau\}$ [$i = m + 1$ in (2)]
17) $d_m = t_m + (S_m/r_m)$ [$i = m$ in (3)]
$\leq t_m + D + (m-1)\tau - t_m$ [$i = m$ in (5), step 14]
$= D + (m-1)\tau < m\tau + D$
18) $(m+K)\tau < m\tau + D$ [by (1)]
19) $t_{m+1} < m\tau + D$ [steps 16, 17, and 18]
20) $d_m \geq t_m \geq (m+K)\tau$ [case of $t_m \geq (m+K)\tau$]
21) r_m^U is well defined [case of $t_m < (m+K)\tau$]
22) $d_m = t_m + (S_m/r_m)$
$\geq t_m + (m+K)\tau - t_m$ [$i = m$ in (6), step 21]
$= (m+K)\tau$
23) $t_{m+1} = d_m$ [$i = m + 1$ in (2), steps 20 and 22]

Steps 15, 19, and 23 demonstrate that (7), (8), and (9) hold for $i = m$. Proof of induction step is complete. □

Acknowledgment

The authors thank T. Y. C. Woo for technical comments and help in the preparation of figures. The authors greatly appreciate constructive comments by the anonymous referees.

References

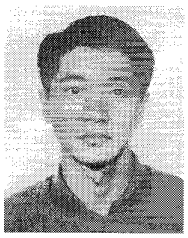
[1] F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," *IEEE Network*, pp. 25–39, Mar./Apr. 1995.

[2] L. Delgrossi *et al.*, "Media scaling for audiovisual communication with the Heidelberg transport system," in *Proc. ACM Multimedia'93*, Aug. 1993, pp. 99–104.
[3] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *CACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991.
[4] K. Joseph and D. Reininger, "Source traffic smoothing for VBR video encoders," in *6th Int. Workshop Packet Video*, Portland, OR, Sept. 1994.
[5] H. Kanakia, P. Mishra, and A. Reibman, "An adaptive congestion control scheme for real-time packet video transport," in *Proc. SIGCOMM'93*, Sept. 1993, pp. 20–31.
[6] S. S. Lam, "A model for lossless smoothing of compressed video," in *Experts on Networks Symp.: 60th Birthday of Professor Leonard Kleinrock*, UCLA, June 1994.
[7] S. S. Lam, S. Chow, and D. K. Y. Yau, "An algorithm for lossless smoothing of MPEG video," in *Proc. ACM SIGCOMM'94*, London, England, Aug. 1994.
[8] T. Ott, T. Lakshman, and A. Tabatabai, "A scheme for smoothing delay-sensitive traffic offered to ATM networks," in *Proc. INFOCOM'92*, 1992, pp. 776–785.
[9] P. Pancha and M. El Zarki, "Bandwidth requirements of variable bit rate MPEG sources in ATM networks," in *Proc. INFOCOM'93*, Mar. 1993, pp. 902–909.
[10] C. Partridge, *A Proposed Flow Specification*, Internet RFC 1363, Sept. 1992.
[11] A. Reibman and A. Berger, "On VBR video teleconferencing over ATM networks," in *Proc. GLOBECOM'92*, 1992, pp. 314–319.
[12] D. Reininger, D. Raychaudhuri, B. Melamed, B. Sengupta, and J. Hill, "Statistical multiplexing of VBR MPEG compressed video on ATM networks," in *Proc. INFOCOM'93*, Mar. 1993, pp. 919–925.
[13] D. K. Y. Yau and S. S. Lam, "Operating system techniques for distributed multimedia," *Technical Report TR-95-36*, July 1995. Available from http://www.cs.utexas.edu/users/lam/NRL; an early version in *IS&T/SPIE Proceedings Multimedia Computing and Networking Conference*, Jan. 1996.

Simon S. Lam (S'71–M'74–SM'80–F'85) received the B.S.E.E. degree (with Distinction) from Washington State University, Pullman, in 1969, and the M.S. and Ph.D. degrees in engineering from the University of California at Los Angeles, in 1970 and 1974, respectively.

From 1971 to 1974, he was a Postgraduate Research Engineer at the ARPA Network Measurement Center (UCLA). From 1974 to 1977, he was a Research Staff Member at the IBM T.J. Watson Research Center, Yorktown Heights, New York. Since 1977, he has been on the faculty of the University of Texas at Austin, where he is a Professor of Computer Sciences. He holds two anonymously endowed professorships, and served as Department Chair from 1992 to 1994. His research interests are in network and protocol design, performance analysis, distributed multimedia, and security. He served on the Editorial Boards of *Performance Evaluation*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON COMMUNICATIONS, and the PROCEEDINGS OF THE IEEE. He organized and was Program Chair of the first ACM SIGCOMM Symposium held at the University of Texas at Austin in 1983. He presently serves as Editor-in-Chief of the IEEE/ACM TRANSACTIONS ON NETWORKING.

Dr. Lam received the 1975 Leonard G. Abraham Prize Paper Award from the IEEE Communications Society for his paper on packet switching in a multiaccess broadcast channel, derived from his doctoral dissertation.



Simon Chow received the M.A. degree in computer sciences from the University of Texas at Austin in 1994

He is presently an Applications Developer at the Oracle Corporation, Redwood Shores, CA.

David K. Y. Yau received the B.Sc. degree (with first class honors) from the Chinese University of Hong Kong and the M.S. degree from the University of Texas at Austin.

From 1989 to 1990, he worked in the Distributed Computing Group of Citibank N.A. He is presently a Ph.D. candidate in the Computer Sciences Department at the University of Texas at Austin. His current research interests are in operating system and networking support for distributed multimedia.