*Real-Time Imaging* 7, 000–000 (2001)
doi:10.1006/rtim.2001.0229, available online at http://www.idealibrary.com on IDEAL®

# *PCRTT Enhancement for Off-Line Video Smoothing*

An enhancement of the Piecewise Constant Rate Transmission and Transport (PCRTT) algorithm for reducing the burstiness of a video stream based on smoothing constant intervals is proposed. The new algorithm, called e-PCRTT, relies on geometrical considerations rather than traditional rate-control analysis. E-PCRTT is shown to construct transmission rate-plans with smaller buffer sizes, as compared to the original PCRTT, and alternatively, for the same buffer size, e-PCRTT reduces the number of bandwidth changes compared to PCRTT. In addition, e-PCRTT produces a rate-plan that requires a smaller initial playback delay.

**Ofer Hadar[1] and Reuven Cohen[2]**

[1]*Communication Systems Engineering Department, Ben-Gurion University of the Negev, Beer-Sheva, 84105 Israel E-mail: hadar@bgumail.bgu.ac.il; Tel: +972-7-6477233; Fax: +972-7-6472883*

[2]*Department of Computer Science Technion, Haifa 32000 Israel*

## Introduction

The transfer of pre-recorded compressed video requires the network to support large fluctuations in bandwidth on multiple time scales. Bandwidth smoothing techniques are often employed for reducing the burstiness of a pre-recorded compressed video stream by prefetching frames into the client playback buffer. Using these techniques, a variable bit rate (VBR) stream can be represented as a series of fixed constant bit rates (CBR), thus simplifying the allocation of network resources and increasing their utilization. The smoothing process and the temporary accumulation of data stream in a client buffer before playback increase the advantage of multiplexing. This is because they remove the serial correlation produced by the variable length entropy coders and the smoothing of the short-term sub-image bit-rate variabilities [1]. Smoothing the traffic before launching packets into the network can also increase network reliability since the packet loss behavior of a network is strongly dependent on its workload. For example, if most traffic sources are *smoothed*, that is the ratio of the standard deviation to its mean in their transmission rate is small, then congestion is uncommon and the packet loss rate becomes smaller [2]. As the traffic becomes burstier, the loss rate increases for the same network utilization and buffer availability because transient overloads occur more often.

The process of smoothing the traffic is referred to as *traffic shaping*. In this paper, traffic shaping is implemented by an off-line plan design and a client buffer. The resulting stream is "less bursty" than the original stream in the sense that one of the following holds: (1) the peak rate to the average rate ratio is decreased; (2) the peak rate to the minimum rate ratio is decreased;

(3) the standard deviation rate to the average rate is decreased. For a given client prefetch buffer size, several algorithms for bandwidth smoothing have been introduced and shown to be optimal under certain constraints. Based on a priori knowledge of frame lengths, these algorithms can significantly reduce the burstiness of resources required for pre-recorded video, transfer, and playback [3–6].

This paper focuses on a smoothing technique, known as the *Piecewise Constant Rate Transmission and Transport* (PCRTT) algorithm [5]. This algorithm divides the video stream into fixed-size intervals thus creating a *bandwidth allocation plan*. The main advantage of this method over other methods is that for small buffer sizes, PCRTT creates bandwidth plans that have near optimal peak bandwidth requirements, while requiring very little computation time [3]. Since a PCRTT plan consists of fixed-size intervals, the bandwidth changes occur after constant times. This can be useful for the multiplexing of several streams according to the algorithm described later. Another advantage of PCRTT is that it can produce bandwidth plans with a meaningful lower bound on the minimum time between rate changes.

In this paper, we propose an enhancement to PCRTT, which we call e-PCRTT. e-PCRTT aims at reducing the required buffer space for the same fixed-size interval. Alternatively, e-PCRTT can increase the length of the smoothing interval while using the same buffer size. The practical implication of increasing the interval size is that the number of bandwidth changes is reduced. This is of importance for networks such as ATM, which allow users to renegotiate their traffic parameters. Renegotiation of traffic parameters usually requires the end host to send a signaling message along the data path with the new traffic parameters [4]. If the request is feasible, the network signals the host to start sending traffic according to the new parameters.

The idea behind e-PCRTT is to impose local constraints on the trajectory bandwidth rate-line instead of the global constraints imposed by the original algorithm. With PCRTT, image quality degradation is attributed either to decoder buffer overflows, when the encoded data is transmitted too fast into the network, or to decoder buffer underflows, when data is not transmitted fast enough into the network. Throughout the paper, we consider only the end nodes behavior, while assuming the network provides the QoS required by the algorithm, namely *guaranteed bandwidth* and *no jitter*. Hence, data loss is only attributed to decoder buffer overflow and not to network congestion.

The simulations in this paper are based on Motion-JPEG (M-JPEG) encoded video streams that have been generated by W. Feng [3]. In M-JPEG, each video frame is compressed independently, and the traces do not capture the effects of inter-frame dependencies that exist in MPEG-encoded streams. For a typical video source, MPEG encoding has smaller average frame sizes and larger short-term burstiness, due to the mixture of interpolated (I), predictive (P), and bidirectional (B) frames [7]. These three different types of frames are grouped as units of Group-of-Pictures (GOPs), where each GOP consists of an arrangement of one I-picture, P-pictures, and B-pictures. The burstiness within a GOP can be resolved by prefetching the short-term variation into a relatively small client buffer size. As indicated in [3], the relative performance of bandwidth smoothing algorithms is more sensitive to the medium-term and long-term burstiness in the underlying video stream, particularly for a larger client buffer. Since a real-time MPEG encoder would not significantly affect the performance trends, except perhaps under small buffer sizes, there is no justification for using MPEG hardware which is in order of magnitude more expansive than the implementation by M-JPEG hardware [3]. Nevertheless, the e-PCRTT presented in this paper can be also effective for removing the short-term burstiness in MPEG video, as well as the medium-term burstiness within and between scenes. The e-PCRTT breaks a long MPEG sequence into several short segments where each segment consists of multiple GOP and calculates a constant bandwidth that is required during each segment such that the playback buffer does not overflow or underflow. By choosing the interval size to be equal to the number of frames in a GOP, it reduces the short-term variations, while for larger interval sizes it can reduce the medium-term variations. One work [8] has described an approach for shaping and smoothing traffic from VBR MPEG video encoders and interfacing to ATM networks by introducing delay and rate buffering between the video encoding and decoding processes, and by flow controlling the data rate from the encoder buffer into the network. The objective of this algorithm is to achieve uniformity of the traffic over the entire MPEG GOP in order to prevent bursts in the traffic profile related to occurrence of I-frames or P-frames.

Another possibility for reducing the fluctuations of a compressed video stream is by using adaptive video techniques. In these techniques, the encoder gets a feedback from the channel and adapts all its compression parameters to the actual channel characteristics. The main disadvantage of these techniques is that they change the video quality during playback. In contrast to video rate smoothing algorithms, the quality is not affected by the smoothing process as long as the transmission rate-plan does not produce client buffer underflow or overflow. In addition, unlike most of the adaptive video techniques, real time intensive computation is not necessary.

The rest of the paper is organized as follows. In the next section, PCRTT and other common smoothing techniques for reducing the burstiness of video streams are described. To follow, we present the new proposed e-PCRTT algorithm. The next section compares PCRTT and e-PCRTT. The penultimate section proposes an algorithm for efficient multiplexing of several video streams, and lastly we conclude the paper.

## Bandwidth Smoothing Techniques

The main problem of transmitting compressed video over communication networks is its burstiness. Compressed movies exhibit peak rates that are often significantly larger than their long-term average rate [4]. One of the suggested solutions for reducing burstiness is to use a playback buffer at the client site (PC, workstation, or set-top box). In such a system, the video streams are stored at a multimedia server and transmitted through the network to the client site. The client's video card then decodes the stream and forwards it to the video display. The server can significantly reduce the bandwidth requirements for transmitting stored video streams by pre-fetching frames into the client playback buffer. In order to compute an efficient server transmission plan, bandwidth smoothing algorithms have usually required a priori knowledge of the client buffer size and the length of the transmitted video frames [6].

The system described above provides the client with the possibility to reserve bandwidth during connection set-up. After the connection is established, the bandwidth may or may not be renegotiated, depending on the interface supported between the host and the network [12]. The client buffer functions as a temporal storage for early arriving frames. The smoothing algorithm should produce a transmission plan that minimizes buffer overflow and underflow. As already explained, buffer underflow occurs when the buffer is empty and the decoder has nothing to decode, whereas buffer overflow occurs when frames need to be dropped due to lack of space at the receiver buffer. More specifically, consider a video stream with $N$ frames, where frame $i$ is $x_i$ bytes long, and a smoothing buffer size of $B$ bytes. In order to avoid buffer underflow, the server must always transmit more data than the decoder consumes. Hence, by the time the client decodes the $k$'th frame, $k = 0, 1, 2, \quad N$, it must have received at least $L_k$ frames from the server, where:

$$L_k = \sum_{i=1}^{k} x_i. \tag{1}$$

In the same way, an upper limit on the amount of data the receiver can receive at the time when the $k$'th frame is decoded is given by:

$$U_k = B + \sum_{i=1}^{k} x_i. \tag{2}$$

The two functions $L_k$ and $U_k$ are equidistant functions that create a "river" that delimits the server bandwidth plan. The goal of any smoothing algorithm is to create a rate-plan with a piecewise linear path that stays between $L_k$ and $U_k$. In this paper, we show how such a path can be created using geometrical considerations rather than traditional rate control analysis. In order to avoid overflow and underflow of the receiver buffer, the sequence of transmission rates $r_1, r_2, \quad, r_M$, referred to as the transmission plan or simply transmission, must satisfy:

$$L_k \leq \sum_{i=1}^{k} r_i \leq U_k, \quad (1 \leq k \leq M), \tag{3}$$

where $r_i$ is the transmission rate during the $i$'th time interval and $M$ is the number of intervals. Bandwidth smoothing algorithms typically select the starting point for interval $j+1$ based on the trajectory for interval $j$. By extending the fixed rate line for interval $j$, the trajectory eventually encounters the underflow curve, the overflow curve, or both, requiring a change in the server transmission rate [3]. In the rest of the paper, the two equidistant functions that bound the "river" are referred to as the $L$ curve and the $U$ curve.

Several algorithms have been proposed for resolving the problem of bandwidth smoothing [13–16]. Many of them are based on different criterion for optimality. For example, the *critical bandwidth allocation* (CBA) algorithm [13,14] has the minimal number of bandwidth increases, and the smallest peak bandwidth requirement.

An improvement of this algorithm, the *minimum changes bandwidth allocation* (MCBA) algorithm [15] minimizes the number of rate decreases. In [16], an algorithm is developed to reduce the variability in the rate requirements across the lifetime of the transmission plan. This approach is known as the *minimum variability bandwidth allocation* (MVBA) algorithm [3].

Another smoothing algorithm is presented in [18]. This algorithm is shown to be optimal in the sense that it achieves the greatest possible reduction in rate variability when sending stored video to a client with a given buffer size. However, this algorithm is not optimal for the case where rate changes can only occur at specific points of time as in our case. The core of this algorithm is to change the rate at the latest possible point in time in such a way that the necessitated rate change can be as small as possible. Therefore rate changes may occur at any given time.

In this paper, we concentrate on improving the PCRTT algorithm, that creates a transmission plan with fixed-size intervals. The main advantage of PCRTT compared to other smoothing algorithms is that for small buffer sizes, the created plans have relatively small peak bandwidth requirements, while requiring very little computation time [3]. Also, with this method the transmission schedule plan consists of constant intervals. This might be useful for multiplexing multiple video streams that are smoothed according to the same base time (time interval).

PCRTT determines a single run for each time interval by connecting the intersection points of the vertical borderlines of the time intervals with the $L$ curve. The slopes of the $j$'th line corresponding to the rate $r_j$ is the resulting transmission plan. To avoid buffer underflow, PCRTT offsets this plan vertically when needed. This guarantees that the whole run lies above the $L$ curve [3]. Raising the plan corresponds to introducing an initial playback delay at the client site. When an offset that maintains the plan between the $U$ and $L$ curves does not exist, the size of the smoothing interval should be reduced. This increases the number of bandwidth changes during the transmission of the video stream. The resulting transmission curve determines the minimum buffer size needed to avoid overflow for the given interval size. A detailed mathematical description for PCRTT is presented in [5], which includes development of fundamental relationships between the PCRTT transmission rates, the client buffer size, and the initial delay.

## The e-PCRTT Algorithm

A disadvantage of PCRTT is that the derivation of the bandwidth plan is based only on the lower bank river ($L$ curve). The $U$ curve functions only as an upper limit that prevents buffer overflow. However, it is not considered during the process of creating a bandwidth plan. As a result, the required minimum buffer size tends to be large, especially for streams that exhibit high spatial or temporal activity. The new algorithm, e-PCRTT, developed here derives the bandwidth plan while considering both the $U$ and the $L$ curves. e-PCRTT needs to know the number of frames and the buffer size for each interval the video stream is divided into. In order to construct a legal plan, e-PCRTT forces the computed trajectory *to be adjacent to the central path between the* L *curve and the* U *curve*. In order to minimize the possibility of overflow or underflow, the fixed-rate line in each interval is selected to be equally distant from the $U$ and $L$ curves. e-PCRTT's main contribution is thus to improve buffer utilization at the client site. Consequently, the constructed rate-plan has a fewer number of bandwidth changes compared to PCRTT for the same buffer size. Moreover, for a bandwidth allocation plan with the same number of intervals, the new proposed algorithm requires a smaller buffer size while still avoiding buffer underflow or overflow.

According to e-PCRTT, the starting point for beginning a bandwidth plan at the first interval is the middle of the buffer. This point introduces an initial playback delay, which, as shown in the later section on performance comparison is usually smaller than that introduced by PCRTT. e-PCRTT then constructs a triangle with one vertex at the starting point and the other two vertices on the vertical line. The triangle bounds the first interval (See Figure 1). The rate for this interval is chosen to be the median line of the triangle [the bold lines in Figure 1(a)]. The intersection point of this line with the vertical ending line determines the starting point of the new rate for the second interval. In a similar way, the fixed-rate line is constructed for each succeeding interval. The whole process is described schematically in Figure 1(a). In this figure, the dashed lines represent the two borderlines of the triangle, whereas the solid line represents the fixed-line rate for each interval. The two borderlines of the triangle are determined as follows. In the first step, these are the lines that connect the point at the beginning of an interval and the intersection points between the border vertical line, the $U$ curve, and the $L$ curve. However, if these lines cause buffer underflow or overflow, they are

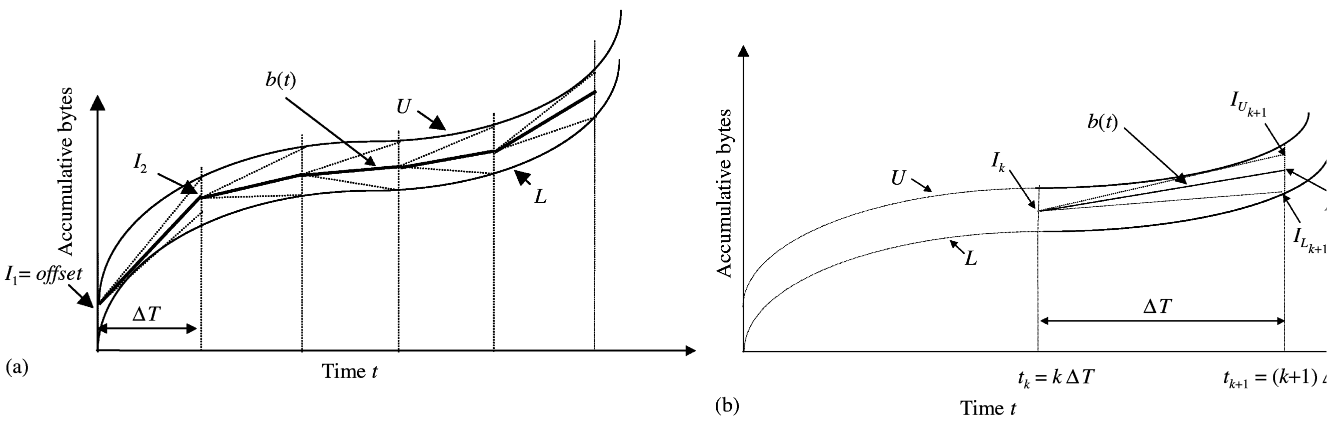

**Figure 1.** The Enhanced PCRTT Algorithm. (a) The construction process; (b) Parameter definitions.

re-defined as the tangent lines with the $U$ curve and the $L$ curve. For cases where the buffer is large enough, most of the triangles are reconstructed according to the first rule. In these cases, the bandwidth allocation plan is coincident with the one created by the original PCRTT algorithm, and the transmission rate-plan created by e-PCRTT is parallel to the one created by PCRTT. The only difference between the two plans is the offset value. However as the buffer size decreases, the triangles become narrower and then they are mostly determined from the tangent lines of the $U$ and the $L$ curves. At the limit, when we continue to decrease the buffer size, the two tangent lines coincide at the most curved interval along the completed path. At this point, the minimum buffer size $B_{min}$ that still eliminates overflow is obtained. Any further decrease of $B$ will cause the triangle to digress out of the river borderlines.

We now define e-PCRTT with more mathematical details. Each deterministic traffic model uses parameters to define a traffic constraint function $b(t)$, which bounds the video server over every interval of length $\Delta T$ (see Figure 1(b)). $\Delta T$ denotes the constant interval that divides the video stream into fixed-size intervals with $\Delta N = \Delta T/F$ frames. It is possible to define constraints for the number of accumulative bytes $I_k$ and $I_{k+1}$ at the beginning and the end of the $k$'th interval respectively:

$$\sum_{i=1}^{k} x_i \leq I_k \leq \sum_{i=1}^{k} x_i + B \tag{4}$$

$$\sum_{i=1}^{k+1} x_i \leq I_{k+1} \leq \sum_{i=1}^{k+1} x_i + B \tag{5}$$

The initial playback delay at the client site $I_1$ is equal to the amount of bytes at the starting point of the rate-plan and is given by:

$$I_1 = x_1 + B/2. \tag{6}$$

The e-PCRTT model is defined as a collection of rate-interval pairs $\left\{ \left( r_k, \Delta \right) | k = 1, 2, ..., M \right\}$, such that the constraint function is given by a piece-wise linear function $b(t)$. This function bounds the number of bytes the source transmits in any interval of length $\Delta T$ by a linear function of $t$:

$$b(t) = \frac{(I_{k+1} - I_k)}{T}[(t - (k-1)\ \ T)] + I_k \tag{7}$$

$$= r_k[(t - (k-1)\ \ T)] + I_k, \quad k \cdot \ \ T \leq t \leq (k+1)\ \ T,$$

where $r_k$ represents the rate at the $k$'th interval, and is given by:

$$r_k = \frac{(I_{k+1} - I_k)}{\Delta}. \tag{8}$$

We now define the upper vertex and lower vertex of the triangle at the $k$'th interval as $I_{U_k}$ and $I_{L_k}$, respectively. Therefore, the ending point of the fixed-rate line at the $k$'th interval $I_{K+1}$ is equal to:

$$I_{k+1} = \frac{\left(I_{U_k} - I_{L_k}\right)}{2} \tag{9}$$

This process for deriving the bandwidth plan continues only if the upper vertex $I_{U_k}$ is larger than the lower vertex $I_{L_k}$ such that a legal triangle is created. Once this condition does not hold, there is no possible triangle at the $k$'th interval, due to the small buffer size that does not allow the rate-plan to continue without crossing the $U$ or $L$ curves. An example for deriving a single rate-line

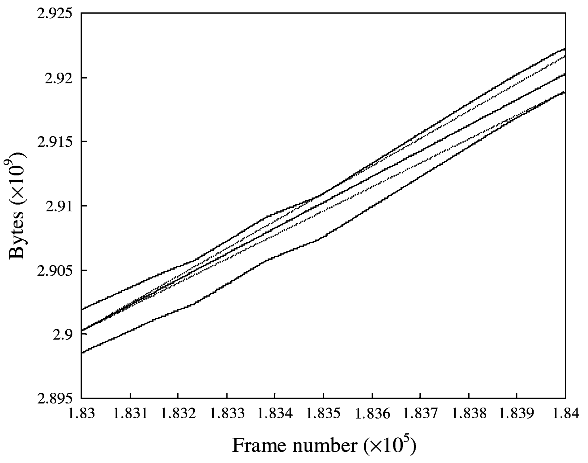

**Figure 2.** An example of a real rate transmission for one interval.

from a real simulation for the video stream "E.T." with interval size of 1000 frames and minimum buffer size of $B_{min}$=504.9 Kbytes is presented in Figure 2. At this specific interval, the triangle is constructed from the tangents of both the lower and the upper curves. This is a typical situation for a curved interval like the one presented in Figure 2.

## Performance Comparison

This section presents a performance comparison of PCRTT and e-PCRTT. This comparison is based on a collection of performance metrics related to the client site that includes the minimum buffer size, the initial playback delay, and the network utilization. We compare the two algorithms by first applying them on an analytical function and then on several investigated video traces. The results of our study show a trade-off between reducing the buffer size and improving the rate performances of the smoothing algorithm. We start this section by comparing the necessary buffer size for an analytical function and then finding the number of bandwidth changes as a function of the buffer size for real video streams. Then, an analytical analysis for buffer utilization is presented. We also present a comparison of the expected initial playback delay by using the two algorithms. Finally, we compare their rate performance.

The experiments in this paper are based on 12 Motion-JPEG video traces that have been presented and discussed in [3]. These traces were downloaded from http://www.cis.ohio-state.edu/$\sim$wuchi. The video library includes clips with different lengths and subjects that are supposed to represent the diversity of compressed video sources in emerging multimedia services. The library includes video with different quality value according to the JPEG standard. Most of the movies considered in this paper have a quantization level that corresponds to quality of a 90 with 0.94 bits-per-pixels. The video stream of "E.T." has the best quantization level: quality factors of 100 correspond to 1.64 bits/pixel. In addition, the library includes three video-recorded seminars, which study the effects of compression and bandwidth smoothing on "educational" video. These seminars were recorded with a single stationary camera focusing on the screen for displaying speaker transparencies. This results in small bandwidth requirements and low variation in the frame sizes as compared to other videos. A table that concludes the statistical characteristics of the video traces is given in [3].

### *Buffer size comparison for an analytical function*

In this section, a comparison between the two algorithms is presented by an analytical function that represents the $L$ curve of the number of accumulative bytes. In this analysis we apply the algorithm on an analytical function, and compare the needed buffer size by the two algorithms. The purpose of our analysis is to show that the minimum buffer size, which is derived by the e-PCRTT, is smaller than that excepted by the PCRTT. For this purpose we select the function $x^n$ where $n$ is an odd integer representing the $L$ curve. In this analysis, the most curved interval is what determines the needed buffer size. For the function $x^n$, this interval is located around the zero axis where the function is transposed from a concave shape into a convex one. Therefore the bandwidth plan is derived around the origin (zero axis), by dividing it into two equal intervals, $[-a, 0]$ and $[0, +a]$, as shown in Figure 3. The minimum buffer size that was derived by the PCRTT algorithm is given by,

$$B_{min\_PCRTT} = 2a^n\left[\frac{1}{n^{\frac{1}{n-1}}} - \frac{1}{n^{\frac{n}{n-1}}}\right], \quad n = 3, 5..., 2n+1, \quad (10)$$

while the minimum buffer size required for e-PCRTT is given by:

$$B_{min\_e-PCRTT} = a^n\left[b \cdot \left(\frac{b}{n}\right)^{\frac{1}{n-1}} - \left(\frac{b}{n}\right)^{\frac{n}{n-1}} - b + 1\right]. \quad (11)$$

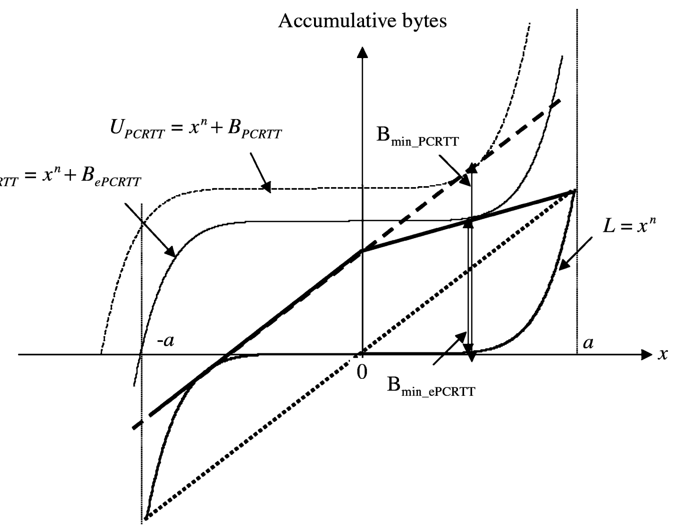

**Figure 3.** Minimum buffer derivation for the analytical function $x^n$, for the two algorithms PCRTT and -PCRTT.

*where*

$$b = 1 - \frac{1}{n^{\frac{1}{n-1}}} + \frac{1}{n^{\frac{n}{n-1}}}.$$

The ratio between $B_{min\_PCRTT}$ and $B_{min\_e\text{-}PCRTT}$ is plotted in Figure 4 as a function of $n$.

As $n$ increases and approaches infinity, the buffer size for PCRTT approaches $2a^n$, while for e-PCRTT it approaches $a^n$. This indicates that the needed buffer size for e-PCRTT is always smaller than for PCRTT. The gap between the two algorithms increases as $n$ increases.

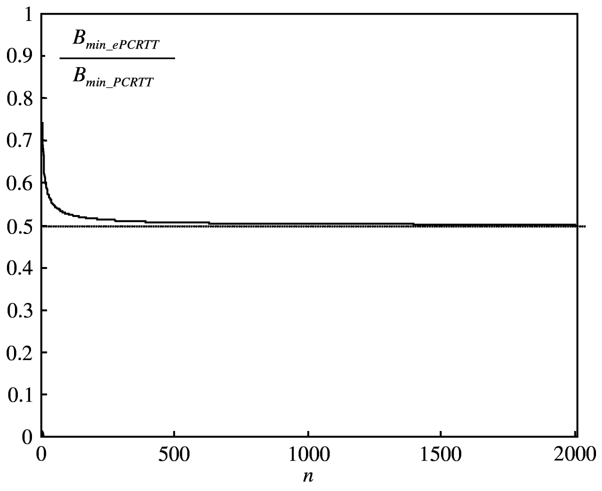

**Figure 4.** The ratio between $B_{min\_PCRTT}$ and $B_{min\_ePCRTT}$ as a function of $n$.

For instance, when $n = 3$, e-PCRTT saves 26% of the buffer space, and as $n$ increases the ratio asymptotically approaches 50% as

$$R = \frac{B_{min\_EPCRTT}}{B_{min\_PCRTT}} \underset{n\to\infty}{\longrightarrow} 0.5 \tag{12}$$

The conclusion that can be derived from the analysis to real video streams is that when the video stream exhibits more burstiness the improvement in buffer size achieved by e-PCRTT increases. This conclusion is supported by the results obtained in the next main section for real video streams.

### *Number of bandwidth changes*

In this section, we expand the analysis presented in the last section to a practical comparison of the two algorithms by applying them to real video streams. However, the main purpose of this comparison is not to measure the interval sizes, but to measure the number of bandwidth changes. These two parameters are inversely related: as the interval size increases, the number of intervals decreases, and vice versa. Our results show that for the same buffer size, e-PCRTT produces a bandwidth plan with fewer rate changes, which implies that the overhead for re-negotiating traffic parameters with the network is reduced, and the advantage of the smoothing process increases.

In what follows, the parameter $M$ will indicate the minimum number of intervals needed for supporting the smoothing process. There is a reciprocal relation between $M$ and the interval duration time $\Delta T$. This relation is represented by $M = T/\Delta$ , where $T$ is the time duration of the video stream. Figure 5 presents the value of $M$ for three streams: "E.T." (100), "Rookie of the Year" (90), and "Seminar-2" (90). Each graph presents the value of $M$ as a function of the buffer size for both PCRTT and e-PCRTT. These results have been obtained by determining a range of realistic buffer sizes and deriving the maximum interval size for each of them. The graphs show that for all buffer sizes, PCRTT requires more bandwidth changes than e-PCRTT. The same results were obtained for the other nine investigated video streams (the graphs are not presented due to limitations of space). Note that the difference between the two algorithms is more noticeable for small buffer sizes.

## *Buffer utilization*

In this section we study the buffer utilization at the client side in order to better understand the results presented previously. We first derive the percentage of the buffer occupancy during the playback of a video stream. Then, the number of bytes at the buffer is determined by subtracting the decoding rate at the buffer output from the smoothing rate-plan $b(t)$ at the buffer input. For every t, the buffer size is given by:

$$Buffer(t = n \cdot \Delta\tau) = b(t) - \sum_{i=1}^{n} x_i, \quad 1 \leq n \leq N \qquad (13)$$

$$where\ 0 \leq Buffer(t = n \cdot \Delta\tau) \leq B_{min}$$

Buffer occupancy can vary during the playback of the video steam from 0 to $B_{min}$. The interesting parameter in this analysis is the *probability density function* (PDF) of the buffer occupancy:

$$PDF = Histogram \times (Buffer(t = n \cdot \Delta\tau)/B_{min},\ 1 < n < N) \qquad (14)$$

Division by $B_{min}$ normalizes the maximum capacity of the buffer to 1. In Figure 6(a), the PDFs of the two algorithms are presented for "E.T." The peak value for the PDF is around 50% for e-PCRTT and 40% for PCRTT. It is also evident that the trajectory path of the rate-plan in e-PCRTT always stays around the middle between the $U$ curve and the $L$ curve. Similar results were obtained for other video streams and for different values of interval size. The meaning of these results is that the bandwidth allocation plan derived by e-PCRTT

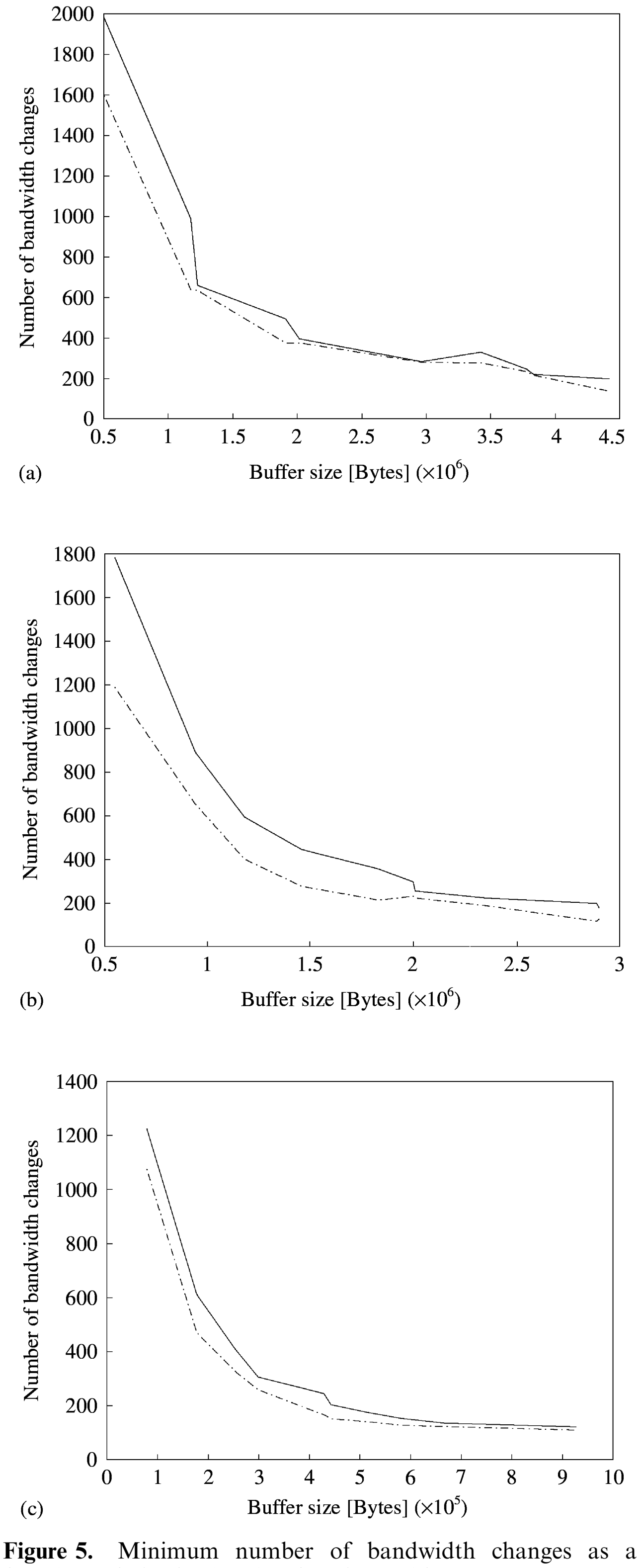

**Figure 5.** Minimum number of bandwidth changes as a function of the buffer size for three video streams. (a) E.T (100); (b) Rookie of the year (90); (c) Seminar-2 (90). —— PCRTT; –·–·– e-PCRTT.

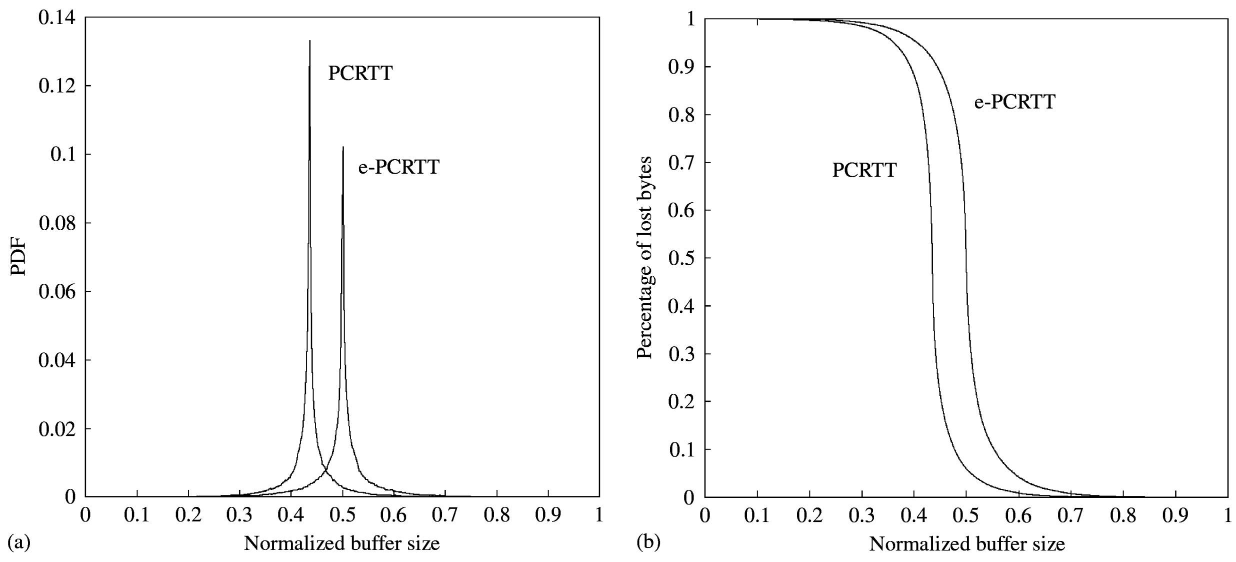

**Figure 6.** Buffer utilization for E.T (100) video trace. (a) Probability Density Function of the buffer content; (b) Percentage of loss bytes.

uses the buffer capacity more efficiently**.** In both methods the minimum buffer size is determined from the peak bandwidth rate in the bandwidth plan, which is generally determined by the maximum frame size. In PCRTT, the buffer size required for transmitting the maximum frame size can be much larger than the buffer size needed for other frame sizes. Therefore, buffer utilization is not as good as in e-PCRTT. This situation especially holds for video streams with high activity and high variability in the original frame sizes. As the original stream is less bursty, buffer utilization can be improved because all intervals demand almost the same buffer size. On the other hand, with e-PCRTT the buffer occupancy always stays steady, at around 50% of the capacity, independent of the interval size or video stream activity.

*Buffer shortage*

We now investigate the effect of reducing the buffer size bellow $B_{min}$, on the percentage of lost bytes due to buffer overflow. The mathematical expression for the percentage of lost bytes can be obtained by integrating the *PDF* between the new normalized buffer size $x$ and 1:

$$P = \int_x^1 PDF(x)dx, \quad (0 < x < 1) \tag{15}$$

Figure 6(a) presents the probability density function of the buffer content for the two algorithms and Figure 6(b) presents the integration results as function of the normalized buffer size when $x$ ranges between 0 and 1. From Figure 6(b) it is evident that for the same normalized buffer size, the percentage of lost bytes due to buffer overflow in e-PCRTT is higher than in PCRTT. This is another indication of the better buffer utilization of e-PCRTT. Recall, however, that the size of $B_{min}$ in e-PCRTT is smaller than in PCRTT. Therefore, the normalization factor in the two algorithms is not the same. We can conclude that, in general, e-PCRTT allows us to use a smaller buffer size, but compared to PCRTT any reduction below this value can cause more losses.

*Initial delay*

Due to the offset of the transmission plan, both PCRTT and e-PCRTT impose an initial playback delay. To ensure constant quality, the data of the first frames is stored in the client buffer prior to playback beginning. The smoothing algorithm usually tries to minimize this delay. Recall our assumption that the network introduces no losses or jitter. Hence, we consider only the delay incurred at the client site due to the transmission plan. The initial playback delay of PCRTT due to raising the plan above the $L$ curve is given by:

$$\sum_{i=1}^{d} x_i = x_1 + \mathit{offset} \tag{16}$$

where *offset* represents the minimum delay that prevents underflow (see Figure 1) and $d$ is the number of frames that satisfies Eqn (16). In e-PCRTT, the initial delay, caused from starting the bandwidth plan at the middle of the buffer size, is equal to:

$$\sum_{i=1}^{d} x_i = x_1 + B/2. \tag{17}$$

By solving Eqn (17) for *d*, the delay is obtained. Note that there is a direct dependency between the initial delay and the buffer size.

The results from the delay analysis of three representative video streams ("E.T.", "Rookie of the Year" and "Seminar-2") are presented in Figure 7. These results were obtained for the same interval size, and for the minimum buffer size each algorithm requires. The graphs show the delay as a function of the interval size. For both algorithms, the initial delay increases when increasing the interval size. Similar results were achieved in [17]. However, the initial delay in e-PCRTT is smaller than in PCRTT for most of the investigated video streams. For example, e-PCRTT reduces the delay of "Seminar-2" by 84% from 19 frames to only three frames as a result of the smaller buffer size required to construct the bandwidth plan in e-PCRTT, compared to PCRTT.

### *Burstiness reduction*

Effective compression techniques, such as MPEG and motion-JPEG, can substantially reduce the resource requirements for storing and transmitting video streams. However, constant-quality compressed video traffic typically exhibits significant burstiness on multiple time scales, due to the frame structure of the compression algorithm, as well as the natural variations within and between scenes. The burstiness of variable-bit-rate traffic complicates the mechanisms that should guarantee proper resource allocation at the network and at the end hosts. In this section, we compare the ability of the two algorithms to reduce the burstiness of the encoded video streams. We consider three parameters that indicate the burstiness of the video stream: *peak-to-minimum ratio* (PMR), *peak-to-average ratio* (PAR), and *average-to-standard-deviation ratio* (ASR) all of which are related to the statistics of the rate plan that were accepted for different values of buffer size after the smoothing process. The smoothing process is expected to reduce the variability of the rate plan and the maximum peak rate. Therefore, PMR and PAR should decrease as the buffer size increases, whereas ASR should increase. In our analysis, we assume an equal

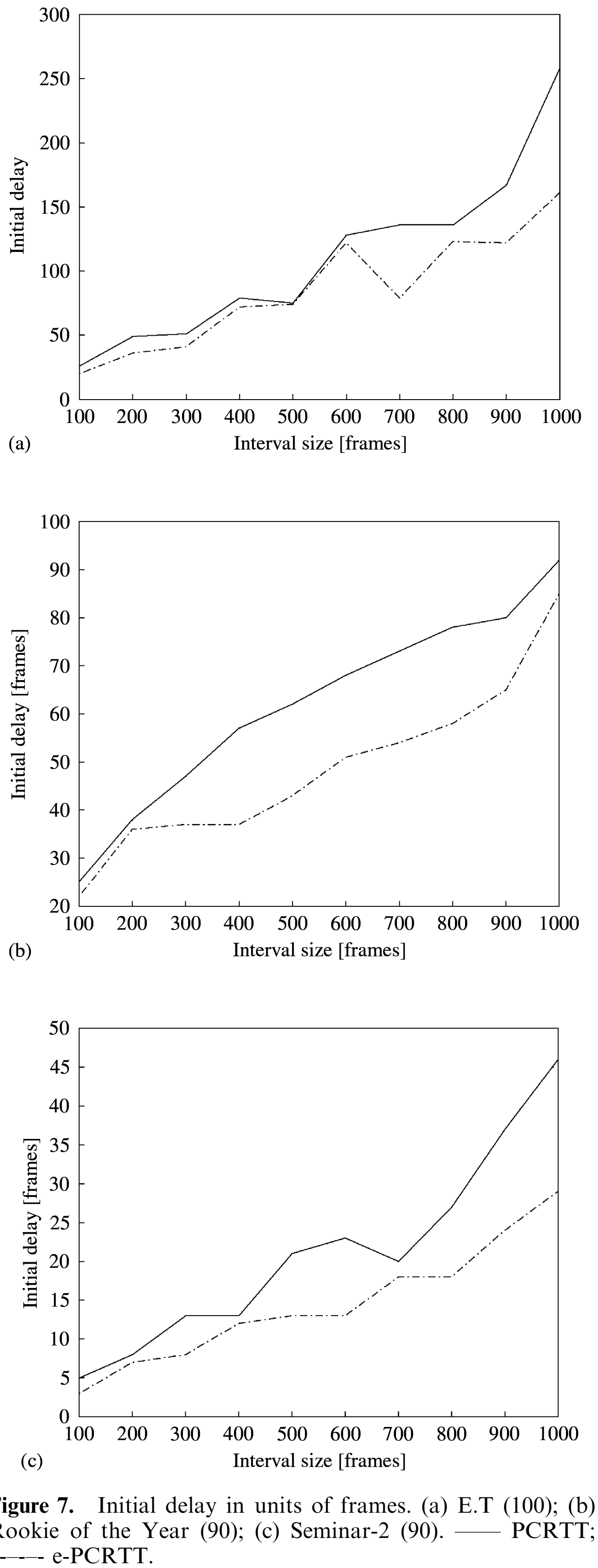

**Figure 7.** Initial delay in units of frames. (a) E.T (100); (b) Rookie of the Year (90); (c) Seminar-2 (90). —— PCRTT; –·–·– e-PCRTT.

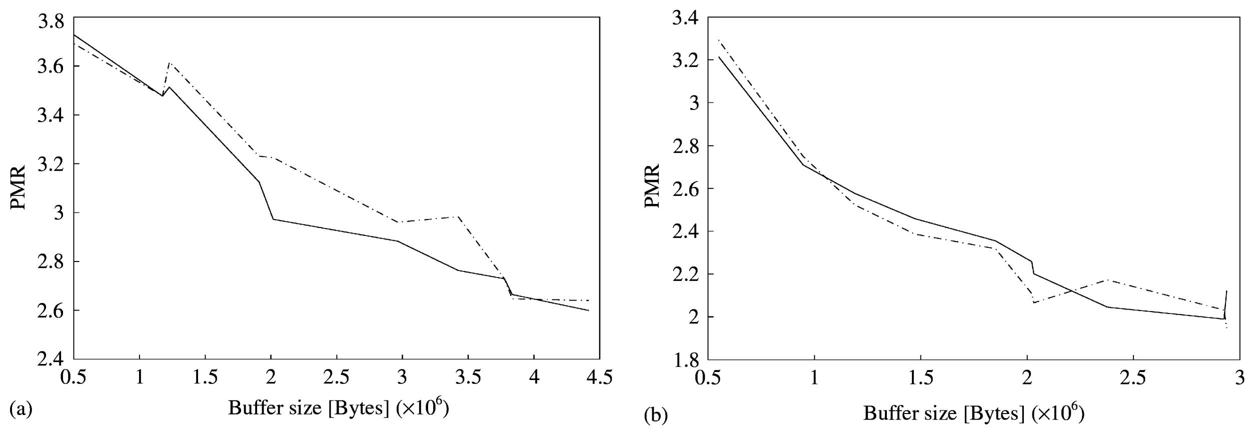

**Figure 8.** Peak-to-Min ratio (PMR) as function of the buffer size. (a) E.T (100); (b) Rookie of the Year (90). —— PCRTT; –·–·– e-PCRTT.

buffer size and different number of intervals for the two algorithms. The number of intervals in each case is chosen to be the minimum that can still guarantee a legal rate plan.

Figure 8 shows the variation of PMR as a function of the buffer size for the two algorithms, in two representative video traces: "E.T." and "Rookie of the Year". As expected, the PMR decreases as the buffer size increases. It is also evident that when the original video trace is more bursty, the reduction of PMR is less noticeable for both algorithms. For "E.T." (100), a 29% reduction in PMR is achieved for a buffer size that varies between 0.5 and 4.5 Mbytes. For "Rookie of the Year" (90), the reduction in PMR is 36% for a buffer size that varies between 0.5 and 3 Mbytes. These results show that e-PCRTT outperforms PCRTT for the "Rookie of the Year" stream, but not for "E.T."

Figure 9 plots the peak-to-average ratio as a function of the buffer size. In this case, since the average rate is not influenced so much from the smoothing process, the duration of the video stream playback is almost the same before and after applying the smoothing algorithm. Therefore, PAR actually represents the amount of peak rate reduction. As expected, the peak rate decreases as the buffer size increases, and it approaches the average rate for a very large buffer size. From this figure it is difficult to decide which algorithm is better, since they both have a similar effect on PAR.

Figure 10 gives a comparison of the variances of the instantaneous bit rates, represented by the ASR

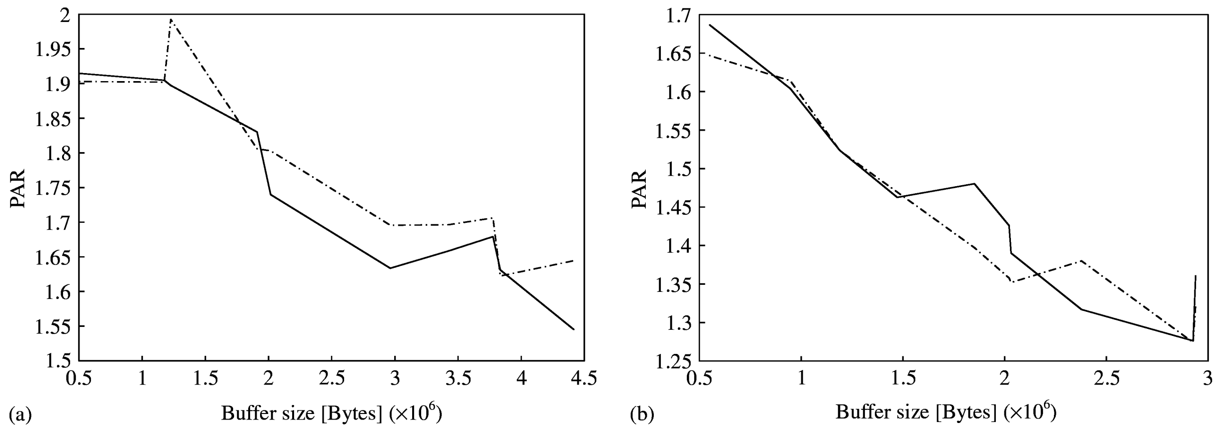

**Figure 9.** Peak-to-Average Ratio (PAR) as function of the buffer size. (a) E.T (100); (b) Rookie of the Year (90). —— PCRTT; –·–·– e-PCRTT.

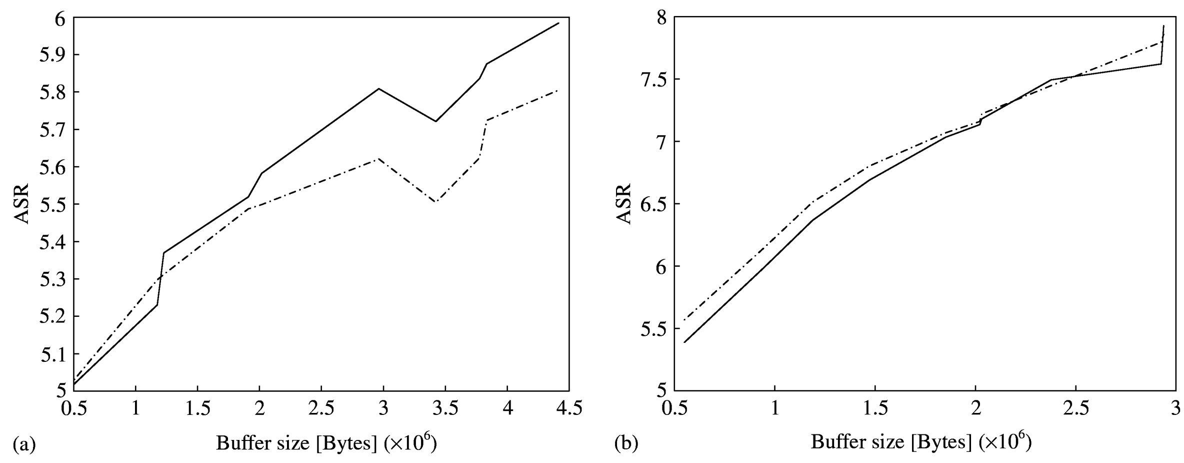

**Figure 10.** Average-to-Std, Ratio (ASR) as function of the buffer size. (a) E.T (100); (b) Rookie of the Year (90). —— PCRTT; –·–·– e-PCRTT.

parameter. In general, the ASR tends to increase as the buffer size increases. Here, PCRTT outperforms the e-PCRTT for most of the buffer sizes for "E.T." (Figure 10a) while e-PCRTT is better for "Rookie of the Year" (Figure 10b).

Based on the results presented in Figures 8–10 and some other results that are not presented in this paper, it is quite difficult to decide which algorithm is more successful in reducing the burstiness of video streams. We expected to get better results for e-PCRTT due to its larger interval size that enable us to pre-fetch more frames into the buffer. However, the actual results show only a small advantage. A possible explanation for this is that in this simulation, e-PCRTT uses the same buffer size as PCRTT but with longer intervals. Therefore, there are some intervals that demand a higher bandwidth rate (especially those intervals that are the most curved at the video streams). The burstiness results are also similar when we use the same number of intervals but with different buffer sizes. In those cases, e-PCRTT produces a rate plan with a smaller buffer size, without increasing the burstiness of the streams.

## Multiplexing of Multiple Streams

One of the main advantages of off-line video smoothing schemes that transform the video stream into a sequence of constant bit rate streams, such as PCRTT and e-PCRTT, is that multiplexing multiple streams into a single CBR channel is facilitated. In the rest of this section, we introduce a possible scheme that increases bandwidth utilization**.** Generally, the main problem with multiplexing is that buffer underflow may occur. However, we can avoid buffer underflow by reducing the rate at intervals where there is a deviation above the peak rate. In those intervals, we can obtain the rate reduction by preventing the rate plan from staying around the middle of the buffer size.

In the general case, we assume that the video streams are smoothed with the same interval size. We further assume that all streams are synchronized, in the sense that all intervals start at the same time instances. The proposed mechanism provides a method to accommodate the bandwidth rate-plan of each multiplexed video stream according to the constraint of the network peak rate.

Figure 11 demonstrates the multiplexing of two original video streams versus the multiplexing of the smoothed versions of the same streams. Notice that multiplexing the unsmoothed streams results in many deviations above the channel rate. As a result, the channel cannot satisfy the required rates for these instances, and packet losses may occur in the network switches. However, when the smoothed streams are multiplexed, deviations above the peak rate occur only at a few intervals. Therefore, we suggest changing the rate-plan of these streams such that there will be no deviation above the peak rate. It is difficult to employ this idea for the original unsmoothed streams, because there are too many discrete deviation points above the peak-rate. The smoothing process permits the rate-plan

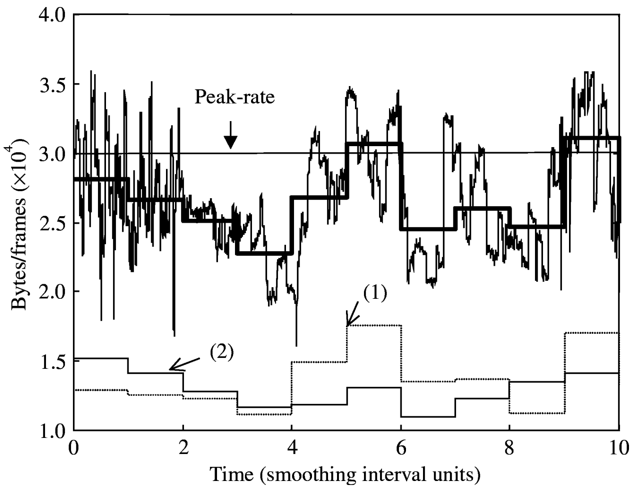

**Figure 11.** An example multiplexing two video streams into a CBR channel before and after bandwidth smoothing. —— Combined smoothed streams; jagged line=combined unsmoothed stream.

to be changed as long as the new rate-plan avoids client buffer overflow or underflow.

The process of reconstructing the rate-plan is shown in Figure 12. Assume that the multiplexed signal consists of $N$ smoothed video streams, and that there is a deviation of $\Delta_{BW}$ above the peak rate at the $j$th interval. In order to avoid this deviation, the transmission rate of each video stream should be reduced for this specific interval. The rate reduction should be performed in a fair manner to all the video streams, such that no stream plan encounters underflow. This is achieved by reducing the rate of each stream $i$ according to the ratio, defined by the rate distance $\Delta r_i$ from the underflow limit divided by the total possible rate reductions. This total rate reduction of all video streams, at the $j$'th interval is defined as:

$$R = \sum_{i=1}^{N} r_i. \qquad (18)$$

Therefore rate reduction for stream $i$ is:

$$_{BW_i} = \frac{r_i}{R} \cdot \; _{BW}. \qquad (19)$$

From Eqns (18) and (19) it follows that the total reduction of all the streams is summed up to exactly $\Delta_{BW}$. The rate reduction at the $j$'th interval requires that the rate-plan of all the affected streams from the $j$'th interval to the last interval be changed. However, the sequence of rate-plans until the $j$'th interval for every stream is correct and therefore should not be changed. In such a way, it is possible to adjust all the rate-plans until there is no deviation above the peak rate for all the intervals.

It is important to indicate that the proposed mechanism does not guarantee full protection from underflow or overflow situations, especially when there is a large deviation above the peak rate. However, it reduces the number of such incidents to a minimum. An improvement of the outcome of this algorithm can be achieved by increasing the client buffer size. This will allow the interval size to be increased and therefore simplify the management process by having a smaller number of

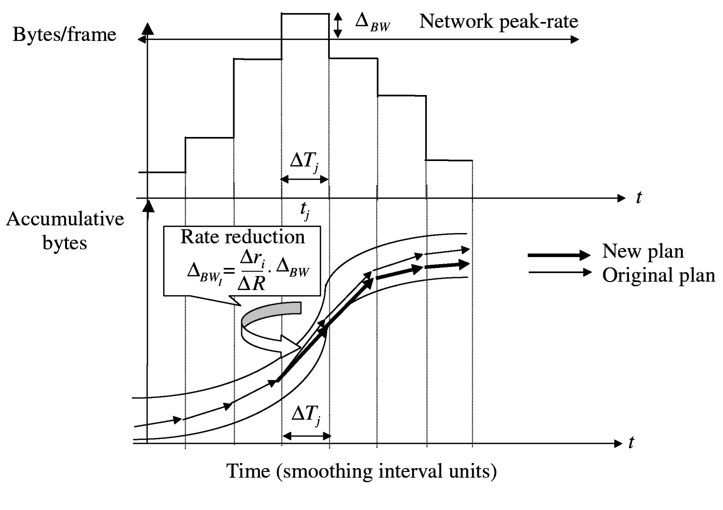

**Figure 12.** The management process of bandwidth rate reduction at the $j$'th interval.

intervals. In addition, the peak rate of each video stream will be reduced such that the total deviation from the channel rate is reduced as well.

## Conclusions

This paper addressed the issue of representing a video stream as a sequence of constant bit rate streams. We have presented an enhancement of the PCRTT algorithm for reducing the burstiness of the compressed stream. The main improvement of the enhancement, called e-PCRTT, is the reduced client buffer as compared to PCRTT. In addition, e-PCRTT produces a bandwidth plan with less intervals (rate changes) for the same given buffer size. This allowed us to reduce the overhead of renegotiation with the network. Another advantage of e-PCRTT is a reduction in the initial playback delay. The proposed algorithm also enhances the profit achieved due to multiplexing several streams into a single channel. Future research may expand the multiplexing mechanism for the sake of minimizing the total transmission cost. Another possible future direction is to expand e-PCRTT for MPEG streams. For such streams the fixed-size interval should be chosen as a multiplication of Group Of Pictures.

## Acknowledgements

We would like to thank Dr. Wu-chi Feng from the Department of Computer and Information Science, Ohio State University for the video streams database.

## References

1. Leduc, J.P. (1994) *Advances in image communication 3 — Digital moving pictures-coding and transmission on ATM networks*. Amsterdam; Elsevier Science.
2. Keshav, S. (1997) *An engineering approach to computer networking: ATM networks, the Internet, and the Telephone network*. IL, U.S.A.: Addison-Wesley.
3. Feng, W. & Rexford, J. (1997) A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video. In *Proc. IEEE INFOCOM, Kobe, Japan, Apr. 1997*, pp. 58–66.
4. Grossglaser, M., Keshav, S. & Tse, D.N.C. (1997) RCBR: A simple and efficient service for multiple time-scale traffic. *IEEE/ACM Trans. Networking* **5**: 741–755.
5. McManus, J.M. & Ross, K.W. (1996) Video on demand over ATM: Constant-rate transmission and transport. *IEEE J. Select. Areas Commun.* **14**: 1087–1089.
6. Rexford, J., Sen, S., Dey, J., Feng, W., Kurose, J., Stankovic, J. & Towsley, D. (1997) Online smoothing of live variable-bit -rate video. In: *Proc. 7th Workshop Network and Operating Systems Support for Digital Audio and Video, St. Louis, MO, May 1997*, pp. 249–257.
7. Gall, D.L. (1991) MPEG: A video compression standard for multimedia applications. *Commun. ACM* **34**: 46–58.
8. Lakshman, T.V., Ortega, A. & Reibman, A.R. (1998) Variable bit rate (VBR) video: Tradeoffs and potentials. *Proc. IEEE* **86**: 952–973.
9. Joseph, K. & Reininger, D. (1995) Source traffic smoothing and ATM networks interfaces for VBR MPEG video encoders. In *Proc. of IEEE INFOCOM, Mar. 1995*, pp. 1761–1767.
10. Pancha, P. & El Zarki, M. (1993) Bandwidth requirements of variable bit rate MPEG sources in ATM networks. In: *Proc. of IEEE INFOCOM '93, March 1993*. pp. 902–909.
11. Kanakia, H., Mishra, & Reibman, A. (1993) An adaptive congestion control scheme for real-time packet video transport. In: *Proceedings of ACM SIGCOMM'93 Sept. 1993*, pp. 20–31.
12. Jiang, Z. & Kleinrock, L. (1998) A general optimal video smoothing algorithm. In: *Proc. of IEEE INFOCOM, March 1998*.
13. Feng, W. & Sechrest, S. (1995) Smoothing and buffering for delivery of prerecorded compressed video. In *Proc. of the IS&T/SPIE Symp. on Multimedia Comp. and Networking, Feb. 1995*, pp. 234–242.
14. Feng, W. & Sechrest, S. (1995) Critical bandwidth allocation for the delivery of compressed video. *Comput. Commun.* **18**: 709–717.
15. Feng, W., Jahanian, F. & Sechrest, S. (1995) Optimal buffering for the delivery of compressed prerecorded video. In: *Proc. of the IASTED/ISMM Int'l Conf. On Networks, Jan. 1995*.
16. Salehi, J., Zhang, Z.-L. Kurose, J. & Towsley, D. (1996) Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. In: *Proceedings of ACM SIGMETRICS, Philadelphia, PA, May 1996*, pp. 222–231.
17. Knightly, E.W. & Rossaro, P. (1995): Effects of smoothing on end-to-end performance guarantees for VBR video. In: *Proc. of the 1995 International Symposium on Multimedia Communications and Video Coding*, 1995.
18. Salehi, J.D., Zhang, Z., Kurose, J. & Towsley, D. (1998) Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing. *IEEE/ACM Trans. Networking* **6**: 397–410.

AUTHOR QUERY FORM

**HARCOURT PUBLISHERS**

JOURNAL TITLE: RTIM    DATE: 4/4/2001
ARTICLE NO. : 20000229

*Queries and / or remarks*

| Manuscript Page/line | Details required | Author's response |
|---|---|---|
| | Please check RH line | |