

Dynamic Programming Based Smoothing of VBR Video Traffic

Cam Barker¹, Zixiang Xiong¹, and Anthony Kuh²

March 6, 2002

Abstract

The compression of videos using variable-bit-rate encoding produces constant quality video at the expense of significant bit-rate variability. Intelligent traffic smoothing techniques are needed to support efficient network resource utilization. In this paper, we examine a class of bandwidth smoothing algorithms and propose a novel enhancement that uses dynamic programming (DP) to improve upon their performance. Using a long, constant-quality MJPEG encoded video trace, we evaluate the DP-based smoothing algorithm (DPS) and examine its effects on the network, server, and client resources necessary for the transmission of prerecorded videos.

I. INTRODUCTION

Video applications, such as video-on-demand services, necessitate the utilization of a large amount of network bandwidth and storage space. Unlike live video, there is not a requirement that transmission decisions be made in real time and some initial delay between sender and receiver is tolerable [1]. As the popularity of video-on-demand services increases, much of network traffic will be transmission of data from prerecorded video sources.

Usually, video sources are encoded to reduce storage and bandwidth requirements. Some video is encoded used a method called constant-bit-rate (CBR) encoding, which simplifies network bandwidth allocation. However, CBR coding produces video with varying quality, for the number of bits used to encode each frame must remain the same for every frame, even during periods of fast action or high detail when more bits are needed to represent such frames with more variation. Such variable-quality encoding is not appropriate for many video-on-demand applications [2].

A method called variable-bit-rate (VBR) coding produces video streams with constant quality, and it is well known that for a given bandwidth VBR encoded videos have a higher perceivable quality that do CBR video streams [2] [3] [4]. Yet there is a trade-off for this increase in quality; VBR encoded video streams exhibit significant rate variability. Without intelligent traffic shaping, transmission of VBR video streams would lead to inefficient network bandwidth utilization.

This intelligent traffic management is implemented by a class of algorithms known as bandwidth smoothing algorithms, which exploit the *a priori* knowledge of a prerecorded video stream to control the burstiness of the transmission bandwidth without loss of data [4]-[12]. By allowing for a client-side prefetch buffer, video servers are able to transmit video frames to a client buffer in advance of each burst so that the client can draw upon the buffered frames to display a video at the required bit rate without dramatic increases in the network traffic.

By analyzing the prerecorded video stream, bandwidth smoothing algorithms can compute a *transmission schedule* that, at the expense of some initial delay and client memory, losslessly delivers a constant-quality video in time for seamless playback while simultaneously offering improvements in network utilization. These algorithms can reduce the burstiness of a VBR video stream by prefetching

¹ Cam Barker and Zixiang Xiong are with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843.

² Anthony Kuh is with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822.

data at a series of fixed rates – the transmission schedule – which both simplifies the allocation of resources in video servers reduces bandwidth negotiations with the network [1].

There are several metrics used to evaluate the efficiency of a transmission schedule, such as number of rate changes, transmission rate variability, required client buffer size, number of bandwidth increases, and initial playback delay. Various bandwidth algorithms are capable of optimizing one of these metrics, or alternatively, generating transmission schedules that meet non-optimal bounds on several metrics, but it is difficult to design a single algorithm that performs optimally for several metrics.

Comparison studies have been performed to evaluate the performance of bandwidth smoothing algorithms relative to each other [1]. It is useful to see how each algorithm performs, not only with respect to the metrics it can optimize, but on the metrics that it cannot, for in many actual situations more than one metric is of importance and it is desirable to select for deployment the bandwidth smoothing algorithm that meets near-optimal bounds for many metrics over the bandwidth smoothing algorithm that performs optimally for one metric but poorly for many others.

Recently, a wavelet-based traffic smoothing (WTS) algorithm has been developed that performs reasonably well under several performance metrics [11]. The algorithm allows network traffic smoothing at several resolutions and takes advantage of the self-similar nature of VBR video traffic. Observations of its limitations have motivated us to develop a dynamic programming based smoothing algorithm (DPS) that strives to improve upon the performance of WTS across several metrics by removing its limitations on transmission schedule generation.

We draw on Feng’s library of full-length, constant-quality video clips [14] to compare the performance of WTS, DPS, and other smoothing algorithms using a M-JPEG encoded movie of sufficient length to study the performance of the algorithms.

The paper is organized as follows. In Section II we summarize the PCRTT algorithm, review the WTS algorithm and explore its limitations, and introduce the DPS algorithm. In Section III we describe the DPS algorithm and its properties. In Section IV we present our experimental results, and in Section V we conclude.

II. EXISTING BANDWIDTH SMOOTHING ALGORITHMS

For a given VBR video, let N denote the length of the video in frames, f_n denote the size in bytes of the n^{th} frame, $0 \leq n \leq N-1$, and B the size of the client buffer. Without loss of generality, we assume a discrete-time model with *frame-times* as the unit of time. Any bandwidth smoothing algorithm can be viewed as an algorithm that places rate changes at m points in a video stream; the frames between two rate changes comprise a *run* and are transmitted at rate r_j , $0 \leq j \leq m-1$. Together the m bandwidth runs form a transmission schedule.

In order to avoid client buffer underflow and permit continuous video playback, the transmission schedule must satisfy a lower bound

$$F_{under}(k) = \sum_{i=0}^k f_i$$

that designates the cumulative bytes required by the client at frame-time k , $1 \leq k \leq N$. In order to avoid overflow of a client buffer of size B , no more bytes than

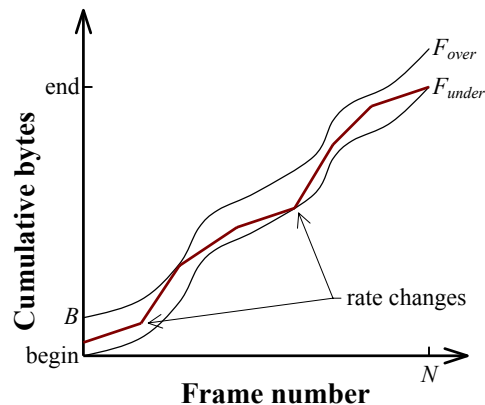


Fig. 1. **Transmission Schedule:** This figure shows for a sample video stream the underflow and overflow curves. The transmission schedule consists of seven runs that satisfy client buffer constraints.

$$F_{over}(k) = B + \sum_{i=0}^k f_i$$

should be transmitted by frame-time k . Any transmission schedule should stay within the bounds set by F_{under} and F_{over} as shown in Fig. 1.

It is common for prerecorded video traces to contain frame quantities on the order of 10^5 , so allowing rate changes after every frame can result in long running times. A heuristic to accelerate running times places the restriction that rate changes may only occur at multiples of L frame-times, in essence dividing the video stream into indivisible *segments* of length L which are transmitted at the average bandwidth requirement for the segment. This heuristic has the added advantage of smoothing the small-scale burstiness occurring as a result of the frame type variation (I, P, B) that occurs in MPEG-encoded prerecorded video streams.

A. MCBA

The minimum changes bandwidth allocation (MCBA) algorithm [7] operates by finding runs that stay within the F_{under} and F_{over} curves dictated by a given buffer size constraint and that minimize the number of rate changes in the transmission schedule. The algorithm is proven to produce transmission schedules with the minimum number of rate changes and the smallest peak rate for a given client buffer size; however, no lower limit can be imposed on the time between rate changes.

B. MVBA

The MVBA algorithm [6] produces transmission schedules that are as smooth as possible for a given client buffer size, as measured by the bandwidth variability metric. Network performance enhancements are explained by the premise that smoother streams require less of network resources. The transmission schedule produced introduces no client playback delay and gradually alters the transmission rate to avoid making large changes that increase rate variability.

C. PCRTT

The piecewise constant-rate transmission and transport (PCRTT) algorithm [4] divides the video stream into segments to create a transmission schedule. First, this $O(N)$ algorithm sets the transmission rate to the average frame size for each segment; each segment corresponds to one run in the transmission schedule. Then the algorithm raises the transmission schedule to avoid client buffer underflow. An initial delay is introduced to the plan so that the client buffer will contain data when playback begins. From the transmission schedule, the algorithm computes the minimum client buffer size to avoid overflow, the maximum distance of the transmission schedule above the underflow curve F_{under} . The PCRTT algorithm forces a rate change after every segment, as shown in Fig. 2(a).

While the PCRTT algorithm operates by raising the transmission schedule above F_{under} , thereby producing a client buffer size requirement, a modification of the PCRTT algorithm called e-PCRTT [12] takes into consideration both the F_{under} and F_{over} curves in addition to a given buffer size constraint. Compared to PCRTT, this algorithm reduces the client buffer size required for lossless playback and produces transmission schedules with fewer number of bandwidth changes.

A further modification of the PCRTT algorithm, PCRTT-DP [9], uses dynamic programming to compute the minimum-cost transmission schedule that contains m rate changes. The cost function can be defined to include many variables, so unlike MCBA and MVBA which calculate an optimal transmission schedule for one metric, PCRTT-DP can produce schedules with pleasing performance across several metrics.

D. WTS

The wavelet-based traffic smoothing (WTS) algorithm [11] calculates a binary tree in which each node represents smoothing at different resolutions. The full tree corresponds to the original video, with each segment of the video stream matching to a leaf node, while a node at a higher level stores one transmission rate for multiple segments of the video. First, this $O(M\log N)$ algorithm builds a binary tree by setting transmission rates for all leaf nodes to the average frame size for each segment; then, in a bottom-up traversal, non-leaf node transmission rates are set to the average rates of each node's children. Next, the algorithm associates a cost with each node at resolution j and offset k using as the cost metric the minimum client buffer requirement when $r_{j,k}$ is set as the transmit rate over frames $[2^j k N, 2^j (k+1) N - 1]$,

$$C_{j,k} = \max_{t \in [2^j k N, 2^j (k+1) N - 1]} \left| \sum_{i=0}^t (r_{j,k} - f_i) \right|.$$

Finally, the WTS algorithm prunes the binary tree to the smallest size that satisfies the constraint requiring the maximum cost of the pruned tree's leaf nodes to be less than the client buffer size. Typically, a node higher in the tree, one which generates a longer run, requires a larger client buffer size to avoid buffer underflow than does one of its child nodes, so a balance is made between runs of greater length and the client buffer size required by a transmission schedule comprised of such runs.

However, the mapping of a prerecorded video stream into a binary tree imposes a limitation on the location of rate changes. Specifically, a single run must have its boundaries on segment numbers that are multiples of powers of two. Since 2^0 equals one, it is possible for runs to start and stop on every interval, but this requires the highest resolution of smoothing, in which case the algorithm degenerates to PCRTT. For every pruning operation that decreases the level of resolution in the binary tree, the boundaries of a run are pushed outward.

Fig. 2(b) shows the results of the WTS algorithm for a sample video stream of four

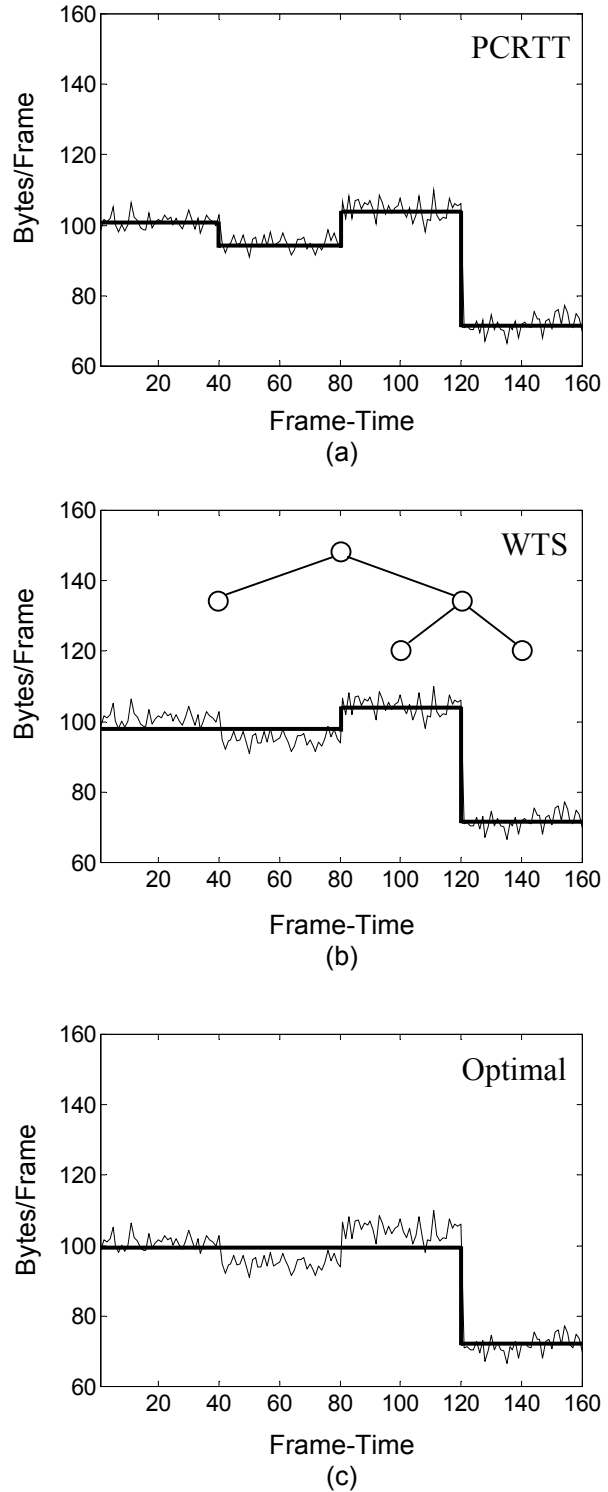


Fig. 2. **Rate change limitations:** These graphs show for sample data how three different bandwidth smoothing algorithms would operate. Segment length is 40 frames. (a) PCRTT forces a rate change after every segment, (b) WTS does not allow a run from the first to third segment unless the fourth segment is in the run as well, but that plan violates the cost constraint so the right subtree is left unpruned, (c) DPS allows rate changes at any time.

segments of 40 frames each. Since the average rate of the fourth segment is significantly different than that of the first three segments, this video stream should have two runs, with the first run consisting of the first three segments and the second run consisting of just the fourth segment. However, when the WTS algorithm is attempted on this video stream, with a reasonable buffer limitation, in order to have the last segment as a separate run, the original video stream has to be split into two halves at the first level of the tree, which results in an inefficient rate change for the given video stream. Such limitations on the placement of rate changes in a transmission schedule, as illustrated by this example, are the motivation for our work on the DPS algorithm.

III. DYNAMIC PROGRAMMING BASED SMOOTHING

A. Motivation

We now introduce a bandwidth smoothing algorithm aimed at relaxing constraints on rate change placement by permitting rate changes at any multiple of L without restriction. The algorithm produces the optimal transmission schedule seen in Fig. 2(c). We assume that the video length N is a multiple of the segment size L , $N = ML$, with M time segments numbered from 0 to $M-1$. The basic idea is to apply dynamic programming techniques, which involves making decisions in stages, where at each stage, the decision is made to minimize a cost function using results from previous decisions.

In our dynamic-programming (DP) based smoothing algorithm, we apply DP to find the optimal transmission schedule for subset of the video stream containing frames $[0, kL-1]$ for $k \in \{1, 2, \dots, M\}$. The power of the DP method is in using known solutions for previous video stream lengths in a manner that avoids an exhaustive search [12]. Fig. 3 provides an explanation of our DP approach. First, we focus on the first two segments of the video stream $[0, 2L-1]$. There are two possible transmission schedules, and finding the winning one with the smaller cost involves little computational effort, as in Fig. 3(a). Then, we consider the optimal segmentation of the first three segments $[0, 3L-1]$. There are four possible ways of placing rate changes into the transmission schedule – no rate changes, at frame L , at frame $2L$, and at both L and $2L$ – but our previous comparison allows us to reduce the comparisons to three options (see Fig. 3(b)) since we know the optimal transmission schedule for the two cases where there is a rate change at frame $2L$.

In general, by utilizing knowledge of $k-1$ previous decisions, we can reduce an exponential search of 2^k possibilities for rate change placement to a linear search of $k+1$ possibilities. Therefore the *DP-based smoothing* (DPS) algorithm has a searching complexity of $O(1 + 2 + \dots + M) = O(M^2) = O(N^2)$. We would not realize this computational savings if the cost function were not separable, that is, once the cost function has been computed for a portion of a transmission schedule $[0, kL]$ any further computation of the cost over $[0, kL]$ is redundant and thus non-optimal transmission schedules can be eliminated efficiently.

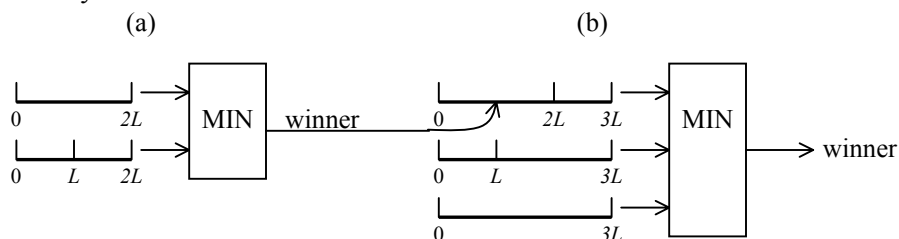


Fig. 3. **Dynamic Programming:** This figure shows the dynamic programming process. (a) First only the portion of the video from $[0, 2L]$ is examined. There are two transmission plans to choose from, with or without a rate change at frame L . The lower cost plan is chosen as the winner. (b) Now the portion of the video from $[0, 3L]$ is examined, but instead of using an exhaustive search to examine four transmission plans, the DP algorithm needs to examine but three.

B. The Algorithm

Let x_k denote a transmission schedule for frames $[0, kL-1]$. As in the WTS algorithm, the cost function C_k associated with a transmission schedule x_k represents the minimum client buffer size required to prevent overflow. The goal of the DPS algorithm involves searching for the best x_M , denoted by x_M^* , that minimizes the cost C_M associated with the whole video stream, frames $[0, ML-1]$.

If we define C_k^* as the minimum cost associated with the best transmission schedule x_k^* for frames $[0, kL-1]$, then the relation

$$(1) \quad C_k^* = \min_{0 \leq t < k} [\max(C_t^*, \Delta C_{t,k})]$$

is satisfied for $k \in \{1, 2, \dots, M\}$ where $\Delta C_{t,k}$ is the minimum cost associated with transmitting at rate $u(j)$ over $tL \leq j \leq kL-1$ and $u(n)$ is the transmission rate at frame-time n . Fig. 4 depicts the above DP computation when $k=M$, C_M^* . Notice that

$$(2) \quad \Delta C_{t,k} = \max_{i \in [tL, kL-1]} \left| \sum_{j=0}^i (u(j) - f_j) \right|$$

is equivalent to

$$(3) \quad \Delta C_{t,k} = \max_{i \in [tL, kL-1]} \left| S_t^* + \sum_{j=L}^i (u(j) - f_j) \right|$$

where $S_t^* = \sum_{j=0}^{kL-1} (u(j) - f_j)$ is a constant inside the \max function. Further, S_t^* is known immediately after C_t^* is computed and since $t < k$ in (1), all S_t^* 's can be retrieved from a lookup table to eliminate redundant calculations in the computation of $\Delta C_{t,k}$.

Our aim is to find the optimal transmission schedule x_M^* for the video stream, and once the minimum costs C_k^* 's have been found for $k \in \{1, 2, \dots, M\}$, we are only one step away from the solution. If we define a backtracking function $b(k)$ of k , $k \in \{1, 2, \dots, M\}$, as

$$(4) \quad b(k) = \arg \min_{0 \leq t < k} [\max(C_t^*, \Delta C_{t,k})]$$

then it is straightforward to recover the optimal transmission schedule through the backtracking relationship, starting from the end of the video stream, $t_{i-1} = b(t_i)$ where t_i is the segment number of the beginning of the i^{th} run of x_M^* , $t_i L$. The DPS algorithm may be summarized as listed in Table 1.

As derived in section II, DPS has a searching complexity of $O(M^2)$ and since it performs M searches, the computational complexity of DPS is $O(M^3) = O(N^3)$.

The cost function in (2) assumes the transmission schedule has been raised to avoid underflow, but the schedule has not, so we must modify the cost function slightly to be a two piece function. The first piece measures the maximum amount by which the transmission schedule dominates F_{under} :

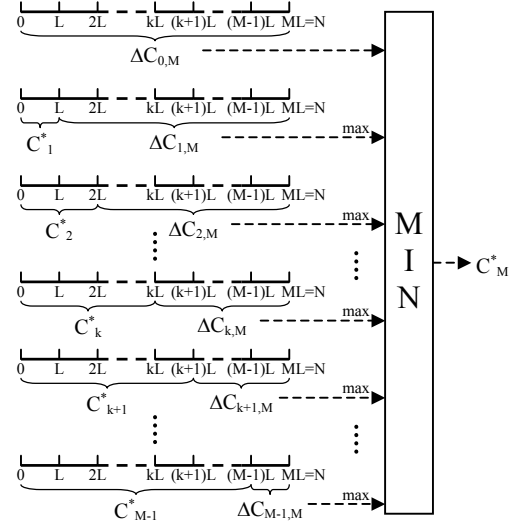


Fig. 4. **DP Computation of C_M^*** : This figure shows the general block diagram of the procedure used to find the optimal transmission schedule cost. The minimization is achieved by the relation $C_k^* = \min_{0 \leq t < k} [\max(C_t^*, \Delta C_{t,k})]$, for $k \in \{1, 2, \dots, M\}$.

TABLE 1
DYNAMIC PROGRAMMING BASED
SMOOTHING ALGORITHM SUMMARY

$$(5) \Delta C_{under,t,k} = \max_{i \in [tL, kL-1]} \left(\sum_{j=0}^i (u(j) - f_j) \right).$$

The second function measures the maximum amount by which F_{under} dominates the transmission schedule:

$$(6) \Delta C_{over,t,k} = \max_{i \in [tL, kL-1]} \left(\sum_{j=0}^i (f_j - u(j)) \right).$$

Implementation of the DPS algorithm remains essentially unchanged, with the cost function $\Delta C_{t,k}$ is comprised of the sum of (5) and (6). Conveniently, $\Delta C_{over_{0,M}}$ is the amount by which the transmission schedule must be raised to avoid client buffer overflow.

- 1) Split the video stream into segments of length L and compute the average rate for each segment.
 - 2) Compute C_k^* as defined in (1) (initialized with $C_0^* = 0$ and $S_0^* = 0$), and record $b(k)$ for $k \in \{1, 2, \dots, M\}$.
 - 3) Extract the optimal transmission plan for the full video stream $[0, ML-1]$, starting from the end of the stream, with the last run being $[b(M)L, ML-1]$, the second to the last run being $[b(b(M))L, b(M)L-1]$, and so forth. The backtracking procedure terminates when $b(t_i) = 0$ for some $i \leq M$.
-

C. Differences between DPS and PCRTT-DP

Another algorithm which employs dynamic programming to calculate optimal transmission rates is the $O(N^3)$ PCRTT-DP algorithm [9]. Similar to DPS, it utilizes DP techniques to calculate optimal transmission rates for a variety of criteria. In recognition of the computational overhead from allowing rate changes after every frame, PCRTT-DP also groups frames into indivisible segments to reduce the algorithm running times to a practical level.

Although similar on the surface, implementation details differentiate PCRTT-DP from DPS. PCRTT-DP calculates the minimum-cost transmission schedule that has k rate changes based on the minimum-cost transmission plan with $k-1$ rate changes, allowing it to minimize receiver memory using a determined number of rates. On the other hand, the DPS algorithm uses dynamic programming to calculate the minimum-cost transmission plan for kL frames of the video based on the minimum-cost transmission schedule for $(k-1)L$ frames, where L is the interval size. Unlike PCRTT-DP, there is no enforced number of rate changes in a transmission schedule generated by DPS. Whereas PCRTT-DP iteratively computes the minimum-cost transmission schedule by adding *rate changes*, DPS iteratively adds *frames* to compute the minimum-cost schedule.

IV. EXPERIMENTAL PERFORMANCE EVALUATION RESULTS

In this section, we compare the PCRTT, WTS, and DPS smoothing algorithms based on widely accepted performance metrics including rate changes per unit time, peak rate requirements, and variability of bandwidth requirements. To evaluate how these algorithms perform over time, we utilize M-JPEG encoded movies from Feng's library. [14] For our comparisons we select a typical 100 minute movie, *Big*, encoded at 30 frames per second and a typical quality of 90, resulting in an average bit rate of 2.96 Mbits per second. By testing performance of smoothing algorithms across a wide range of typical client buffer sizes, we get an accurate measure of how they perform under realistic settings.

For the PCRTT and DPS algorithms, which determine buffer size as a derivative of the transmission schedule, we vary the segment size L to generate a set of schedules, each with a corresponding buffer size. For the sake of a fair comparison we require the WTS algorithm to generate a schedule using the same segment size L with a buffer size no more than the greater of the two buffer sizes associated with the PCRTT and DPS algorithms. By using the same segment size L for WTS, we

are guaranteed that WTS can generate a transmission schedule, since in the worst case WTS produces the same transmission schedule as PCRTT.

For our comparison, it is desirable to find the maximum segment size L for which all three algorithms produce a schedule that will not overflow a given client buffer size, b_0 . For this segment size L , it may be the case that any one transmission schedule requires a smaller client buffer than b_0 , so it is the client buffer requirement of each individual schedule which we associate with an algorithm's performance, not the b_0 buffer size.

For the 181,000 frame movie, the $O(N)$ PCRTT and $O(M\log N)$ WTS algorithms require a few seconds of running time on a 1 GHz workstation. The $O(N^3)$ DPS algorithm using a segment size of 100 requires about 2 hours to execute, though running time drastically decreases as the segment size increases. For example the DPS algorithm executes in about 5 minutes using a segment size of 250 frames and 2 minutes using a segment size of 400 frames.

A. Number of Bandwidth Rate Changes

To make efficient use of network bandwidth, a network manager must track and approve of bandwidth allocation requests. A rate change in a transmission plan causes a negotiation with the network manager in which the manager must provide a guarantee of the requested bandwidth or else reject the request. In doing so the manager needs to consider its current allocations and possibly make a change such as establishing a new route for the transmission or allocating additional bandwidth. Decreasing the number of rate changes in a transmission schedule reduces the cost of negotiating with the network and promotes efficient network utilization [1] [8].

Fig. 5(a) plots the number of rate changes per minute versus the client buffer size. The DPS algorithm results in fewer transmission rate changes than the WTS and PCRTT algorithms for a given buffer size, as expected. For example, transmitting the *Big* video to a client with buffer size of 1MB under the PCRTT algorithm results in 6.2 rate changes per minute. In contrast, the WTS algorithm results in a transmission schedule with 2.4 rate changes per minute, while the DPS algorithm requires 1.3 rate changes per minute.

B. Bandwidth variability

The *coefficient of variation* measures the overall variability in the transmission rate requirements for the video stream [5] [1]. A transmission schedule with a lower coefficient of variation is desirable, for it requires fewer resources from the server and network and lends itself to statistical multiplexing. The DPS algorithm demonstrably offers improvements upon the bandwidth variability of the WTS and PCRTT algorithms, as shown in Fig. 5(b).

C. Peak Rate

The three algorithms under comparison seek to minimize client buffer size, not peak rate, so although the general trend, as expected, is an inverse relationship between client buffer size and peak rate, in some cases the peak rate *increases* with client buffer size. Thus, simply connecting data points on our graph would not convey much useful information.

To illustrate the performance of the three algorithms with respect to minimization of peak rate, a trendline has been fitted to the 164 data points collected for each algorithm, shown in Fig. 5(c). Tightest fit to the data is obtained by using a power series trendline, yielding R^2 values of 0.80, 0.82, and 0.84 for DPS, WTS, and PCRTT respectively. DPS produces the best peak rate minimization, followed by WTS, then PCRTT.

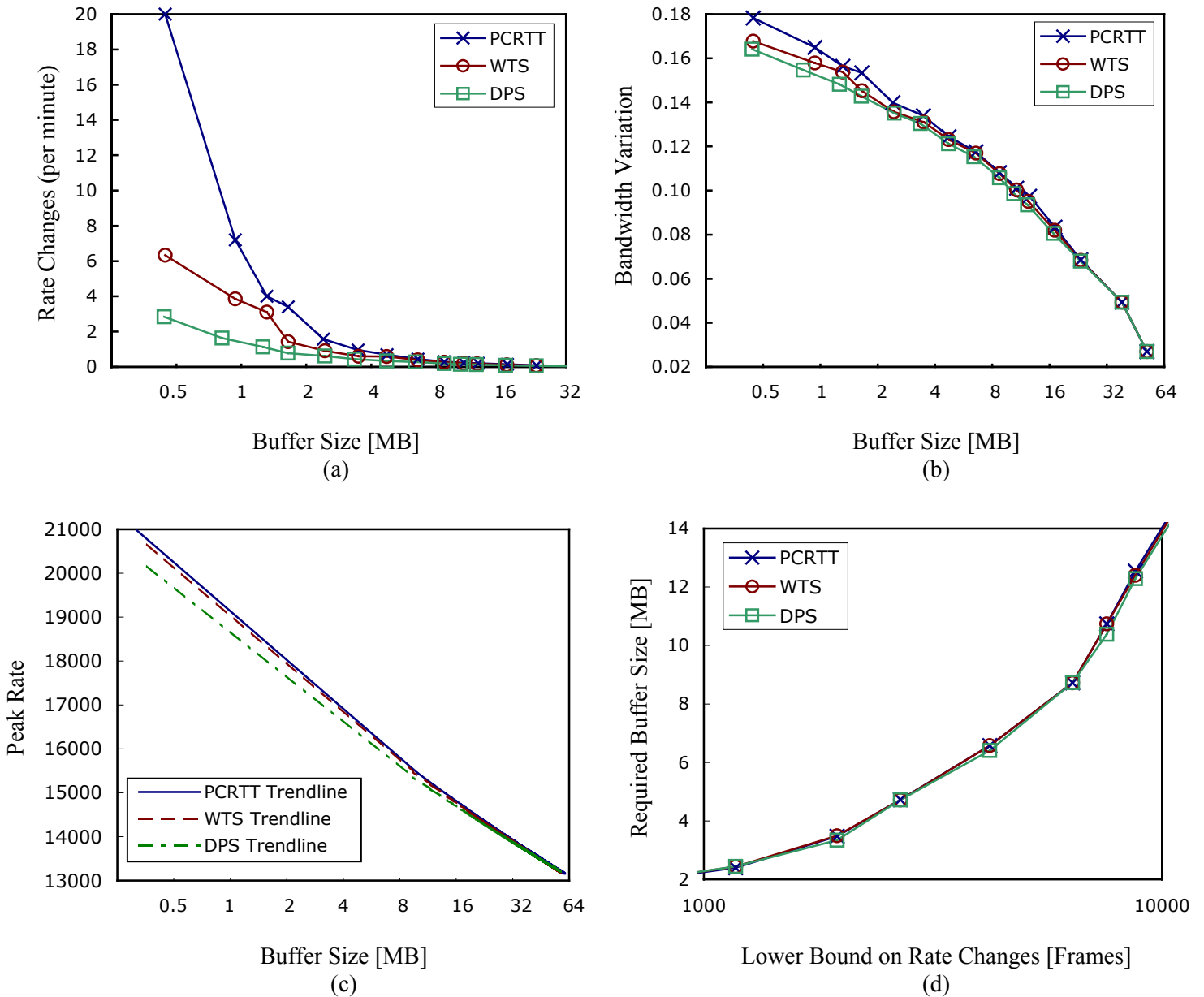


Fig. 5. **Performance Evaluation:** Graph (a) plots the number of rate changes per minute across each algorithm as a function of the client buffer size. The DPS algorithm requires fewer rate changes than WTS and PCRTT. Graph (b) illustrates that DPS has less bandwidth variability than WTS or PCRTT, especially noticeable at small client buffer sizes. (c) Peak rate as a function of client buffer size. Since the peak rate performance is hard to visualize with data points, we use trendlines to observe how well the algorithms perform across a range of buffer sizes. (d) Client buffer size required to guarantee a lower bound on the time between rate changes. Here the algorithms perform similarly, with DPS requiring the smallest buffer size among the three.

D. Lower Bound of Time Between Rate Changes

The segmentation of a video stream into groups of L frames imposes a lower bound on the time between bandwidth rate changes. This property is important in simplifying network resource management, for the network manager avoids reallocating resources at a high frequency [11]. Fig. 5(d) plots the minimum client buffer size required as a function of the lower bound imposed upon a transmission schedule. For a given lower bound, the algorithms require similar buffer sizes, with DPS requiring the lowest amongst the three.

IV. CONCLUSION

In this paper, we have considered the problem of shaping prerecorded video traffic to achieve efficient use of network resources. We examine existing algorithms that enforce a lower bound on the time between rate changes, PCRTT and WTS, and discuss what restrictions they impose on a transmission schedule. Motivated by their limitations on rate change placement, we develop a dynamic programming based smoothing (DPS) algorithm that computes the minimum-cost transmission schedule by iteratively adding frames to a previously optimized video stream.

Experimental results confirm that the DPS algorithm produces transmission schedules that, compared to PCRTT and WTS, have (1) fewer rate changes, (2) less rate variability, (3) lower peak rates, and (4) smaller client buffer size requirements. Given the popularity of applications that require the storage and transmission of compressed video, such as video-on-demand services and digital libraries, such bandwidth smoothing techniques are of importance in efficient network management.

REFERENCES

- [1] W. Feng and J. Rexford, "Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video," *IEEE Transactions on Multimedia*, vol. 1, September 1999.
- [2] A. Reibman and B. Haskell, "Constraints on variable bit-rate video for ATM networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 361-372, December 1992.
- [3] T. V. Lakshman, A. Ortega, and A. R. Reibman, "Variable bit-rate (VBR) video: Tradeoffs and potentials," *Proceedings of the IEEE*, vol. 86, pp. 952-973, May 1998.
- [4] J. M. McManus and K.W. Ross, "Video on demand over ATM: Constant rate transmission and transport," *IEEE J. Select Areas Commun.*, vol 14, pp. 1807-1098, August 1996.
- [5] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," *SPIE Multimedia Networking and Computing*, pp. 316-327, February 1997.
- [6] J. D. Salehi, Z. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 397-410, August 1998.
- [7] W. Feng, F. Jahanian, and S. Sechrest, "Optimal buffering for the delivery of compressed video," *ACM/Springer-Verlag Multimedia Systems Journal*, pp. 297-309, September 1997.
- [8] W. Feng and S. Sechrest, "Critical bandwidth allocation for the delivery of compressed video," *Computer Communications*, vol. 18, pp. 709-717, October 1995.
- [9] J. M. McManus and K.W. Ross, "A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks," *Telecomm. Sys.*, vol. 9, 1998.
- [10] J. Zhang and J. Y. Hui, "Traffic characteristics and smoothness criteria in VBR video traffic smoothing," *Proceedings of IEEE ICMCS'97*, pp. 3-11, 1997.
- [11] D. Ye, Z. Xiong, H. Shao, Q. Wu, and W. Zhu, "Wavelet-based smoothing and multiplexing of VBR video traffic," *Proc. GlobeCom'01*, San Antonio, TX, November 2001.
- [12] O. Hadar and R. Cohen, "PCRTT enhancement for off-line video smoothing," *Real-time imaging*, vol. 7, pp. 301-304, June 2001.
- [13] Z. Xiong, K. Ramchandran, C. Herley, and M. Orchard, "Flexible tree-structured signal expansions using time-varying wavelet packets," *IEEE Transactions on Signal Processing*, vol. 45, February 1997.
- [14] Feng's MJPEG video clip library available at <http://www.cis.ohio-state.edu/~wuchi/Video/MJPEG/>