Multimedia Systems 7: 222–233 (1999)

Multimedia Systems
© Springer-Verlag 1999

Fast techniques for the optimal smoothing of stored video

Sanjay G. Rao⋆**, S.V.Raghavan**

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India
e-mail: sanjay@cs.cmu.edu, svr@iitm.ernet.in

Abstract. Work-ahead smoothing is a technique whereby a server, transmitting stored compressed video to a client, utilizes client buffer space to reduce the rate variability of the transmitted stream. The technique requires the server to compute a schedule of transfer under the constraints that the client buffer neither overflows nor underflows. Recent work established an optimal off-line algorithm (which minimizes peak, variance and rate variability of the transmitted stream) under the assumptions of fixed client buffer size, known worst case network jitter, and strict playback of the client video. In this paper, we examine the practical considerations of heterogeneous and dynamically variable client buffer sizes, variable worst case network jitter estimates, and client interactivity. These conditions require *on-line* computation of the optimal transfer schedule. We focus on techniques for reducing on-line computation time. Specifically, (i) we present an algorithm for precomputing and storing the optimal schedules for all possible client buffer sizes in a compact manner; (ii) we show that it is theoretically possible to precompute and store compactly the optimal schedules for all possible estimates of worst case network jitter; (iii) in the context of playback resumption after client interactivity, we show convergence of the recomputed schedule with the original schedule, implying greatly reduced on-line computation time; and (iv) we propose and empirically evaluate an "approximation scheme" that produces a schedule close to optimal but takes much less computation time.

Key words: Video-on-demand – Bandwidth smoothing – Video compression – Prefetching

1 Introduction

Constant-quality, variable-bit-rate (VBR) compressed video streams can have extremely bursty bit-rate characteristics. This burstiness complicates the task of achieving high resource utilization when such streams are transmitted across a network. Consequently, techniques to minimize the rate variability of transmitted video have been a hot topic of research in recent years. Workahead smoothing is one such method, used in a setting in which a server transmits stored video to a client with a given buffer size. Here, data may be prefetched into the client buffer before its playback time, in a manner that smooths the transmission bit rate. The technique requires that the server compute a schedule of video transfer that ensures the client buffer neither overflows nor underflows.

Several algorithms have been suggested for computing the schedule of transfer for a given video [3, 10, 13] (see [5] for a survey). Briefly, these algorithms assume a pair of constraints $D(t)$,$B(t)$ which respectively denote the minimum cumulative data which the server must transmit by time t to prevent underflow and the maximum cumulative data which the server may transmit by time t without overflow. Each algorithm constructs a piecewise-CBR schedule of transmission $A(t)$ (the cumulative data that the server actually transmits by time t) that satisfies the constraints $D(t) \leq A(t) \leq B(t) \;\; \forall t$, and which is optimal according to some criteria. The algorithms differ from one another with respect to the criterion they optimize. Feng and Sechrest [6] minimize the number of rate increases, Feng et al. [3] the total number of rate changes and McManus and Ross [10] minimize client buffer requirements for a constraint on the total number of CBR transmission segments. In this paper, we base our work on the optimal smoothing algorithm proposed by Salehi et al. [13], which produces a schedule that has minimum peak, variance and rate variability. Section 2 presents an overview of this algorithm. We will refer to the algorithm in [13] as the optimal algorithm and to the resulting schedule as the optimal schedule.

The model assumed by the above smoothing algorithms is *static* in the sense that the constraints D, B are assumed fixed throughout the presentation. The model therefore does not account for dynamic changes (such as changes in client buffer size and worst case network jitter estimates) and client interactions (such as fast-forward and rewind), which alter the work-ahead constraints and consequently the optimal schedule itself. In Sect. 3, we consider a practical setting where dynamic changes and interactions may occur, show

⋆ *Current address*: School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
Correspondence to: S.G. Rao

Table 1. Notation used

N	Length of video in frames
b	Client buffer capacity for storing unplayed frames
$d(t)$	size of frame t
$D(t)$	Minimum cumulative data that must be transmitted by the server by time t to avoid underflow $D(0) = 0$. $D(t) = D(0) + \sum_{i=1}^{t} d(i)$
$a(t)$	Amount of data sent by the server at time t, along the optimal schedule
$A(t)$	Cumulative data sent by the server along the optimal schedule by time t $A(0) = D(0)$. $A(N) = D(N)$. $A(t) = A(0) + \sum_{i=1}^{t} a(i)$
$B(t)$	Maximum cumulative data that can be transmitted by the server by time t without buffer overflow

how the work-ahead constraints are affected and discuss why on-line computation of the optimal schedule becomes essential.

Two fundamental questions that arise in the context of an on-line computation are (i) how to minimize on-line computation time ?, and (ii) what impact does the altered schedule have on the network resources required by the video stream? In this paper, we focus on addressing the first question. Our approach consists of (i) establishing fundamental relationships that exist between the optimal schedules arising out of different constraints (Sect. 4), and (ii) using these results to reduce on-line computation time in the presence of heterogeneous client buffer sizes, variable worst case network jitter estimates and client interactions. (Sect. 5).

In Sect. 4, we assume that two physically different situations may be modeled by the constraints $(D1, B1)$ and $(D2, B2)$ with optimal schedules $A1$ and $A2$. Given particular relationships between $D1$ and $D2$, $B1$ and $B2$, we establish relationships between $A1$ and $A2$. Our results include, (i) the *Domination Theorem* which asserts that, if $D2(t) \geq D1(t)$ $\forall t$ and $B2(t) \geq B1(t)$ $\forall t$, then $A2(t) \geq A1(t)$ $\forall t$, (ii) the *Refinement Theorem* which asserts that, if $D2(t) = D1(t)$ $\forall t$ and $B1$ is displaced downwards or right from $B2$, then the critical points[1] of $A1$ are a superset of those of $A2$, and (iii) the *Convergence Theorem* which asserts that if $\forall t \geq q, D1(t) = D2(t)$ and $B1(t) = B2(t)$, then $A1$ and $A2$ *converge* at a point beyond q.

Applying these results, in Sect. 5, (i) we propose an algorithm that can precompute and store the optimal schedules for *all possible client buffer sizes* in a compact way, (ii) we show that it is theoretically possible to precompute and store the optimal schedule for *all possible worst case network jitter estimates* in a compact fashion, (iii) in the context of client playback after an interaction, we show that the recomputed schedule of transfer *converges* with the original schedule, and recomputation involves examination of only the frames up to the point of convergence. We empirically evaluate the impact of this theoretical result.

Finally, in Sect. 6, we propose and empirically evaluate an *approximation scheme*, that involves altering the work-ahead constraints and performing optimal smoothing with the altered constraints. The resulting schedule is feasible and close to being optimally smooth with respect to the original constraints, yet it takes much less time to compute.

Section 7 discusses related work and Sect. 8 makes concluding remarks.

[1] Critical points of a schedule are those points where the client buffer is empty or full. (See Sect. 4). It is easy to reconstruct the optimal schedule given its critical points.

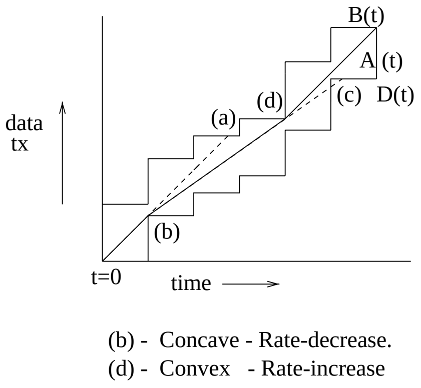


Fig. 1. Optimal schedule construction

2 The optimal smoothing algorithm: overview

Consider a discrete-time model at the frame level. That is, $t \in \{1, 2, \ldots N\}$, where N is the length of the video in frames. The notation is summarized in Table 1. It is assumed that the server transmits data periodically at the video frame rate, and what varies is the amount of data sent each time. $S = [a(1), a(2), \ldots, a(N)]$ represents a feasible schedule iff $A(0) = D(0)$, $A(N) = D(N)$ and $D(t) \leq A(t) \leq B(t)$ $\forall t$. Note that the last condition ensures that there is no buffer underflow ($A(t) \leq D(t)$) or overflow ($A(t) \geq B(t)$). In the particular case where there is no network delay or jitter, and the client has a buffer size b, $B(0) = D(0)$, and $B(t) = \min\{D(t-1) + b, D(N)\}$

The algorithm constructs a feasible piecewise-CBR transmission schedule in the following manner. The idea is to iteratively identify the longest possible CBR transmission segment (since CBR transmission is optimally smooth), and when the transmission rate must be changed to ensure feasibility, to make the change as *early* as possible, thereby ensuring that the change is as *small* as possible. The construction is illustrated in Fig. 1. To prevent buffer overflow at (a), the rate is decreased at the earliest possible moment (b). (Decreasing the rate earlier than (b) would result in a buffer underflow.) Similarly, to prevent buffer underflow at (c), the rate is increased at the earliest possible moment (d). (Increasing the rate earlier than (d) would result in a buffer overflow). The reader is referred to [13] for the actual algorithm. The complexity is $O(N^2)$, but an $O(N)$ algorithm is also discussed. Points of rate increase or rate decrease are called change points, the former being referred to as convex and the latter as concave. Thus, in Fig. 1, (b) is concave and (d) is convex. It is to be noted that if t is a concave (con-

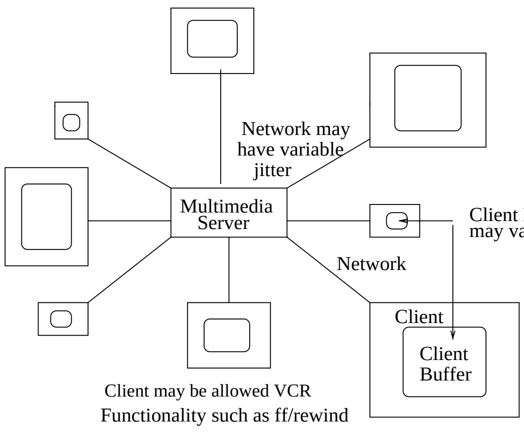


Fig. 2. A typical complex setting

vex) change point, then $a(t+1) < a(t)$ ($a(t+1) > a(t)$) and $A(t) = D(t)$ ($A(t) = B(t)$).

3 Need for on-line computation

Consider Fig. 2 which illustrates a multimedia server, concurrently transmitting distinct VBR streams to clients across a high-speed network. Each client is a set-top box or a network PC with a certain limited buffer capacity, and the server transmits data to the client according to an optimal schedule of transfer, computed as described in the previous section. We now describe several situations where the server needs to compute the schedule of transfer on-line.

In heterogeneous environments, where users purchase set-top boxes corresponding to their budget and requirements, the client buffer sizes can vary from a few hundred kilobytes to over a gigabyte [10]. The work-ahead constraint $B(t)$ varies with client buffer size as $B(t) = \min\{D(t-1) + b, D(N)\}$, resulting in a variation of the optimal schedule with client buffer size. Consequently, the server must either explicitly store the optimal schedule for every possible client buffer size, or it must determine the client buffer size at the start of each presentation and compute a schedule of transfer on-line. Further, the client may be running other processes simultaneously, which share buffer space. This could potentially result in dynamic variation of the buffer available for smoothing during the presentation, and necessitate on-line schedule recomputation.

The client may be connected to the server via a network with highly variable and potentially unbounded jitter, such as Ethernet or the Internet. In [13], it is assumed that the server estimates apriori the worst case end-to-end jitter over the entire length of video playback. A worst case jitter of j frame units, where $j = \lceil \frac{WorstCaseNetworkJitter(s)}{1/framerate} \rceil$, is handled by delaying client playback by j and ensuring that the server stays j ahead of the client, i.e., by shifting the curve $B(t)$ to the right by j units and performing smoothing with the shifted $B(t)$ and the original $D(t)$. (Formally, $B1(t) = B(t-j), t \geq j+1$, $B1(t) = b, 1 \leq t \leq j$, $B1(0) = 0$. Smoothing is done with $D(t)$, $B1(t)$.) However, such an estimate of worst case jitter j is difficult to make in the real environments we consider. Instead, the server could periodically revise its estimate using some on-line algorithm and consequently derive an updated transmission schedule.

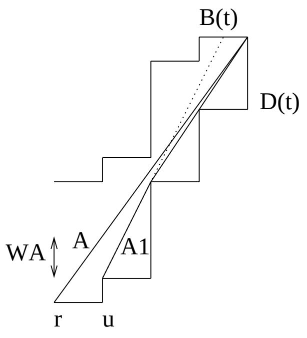


Fig. 3. Playback restart after a client interaction may require schedule recomputation (see text)

Finally, the client may be enabled to perform interactive functions, such as fast-forward, rewind, pause/resume and indexing (temporal jumps forward/backward) during stream playback. While playback restart after an interaction does not alter the work-ahead constraints, a recomputation of the optimal schedule may still be required. To see this, consider Fig. 3, which illustrates a client which while viewing frame r of a video makes a temporal jump forward to frame u. The client buffer is flushed out and normal playback now resumes from frame $u+1$ onwards. Note that had the interaction not taken place, an amount of work-ahead data $WA = A(u) - D(u)$ would have already been available in the client buffer at time u. Consequently, the original schedule cannot be used directly and a new optimal schedule $A1$ needs to be computed.

The optimal off-line algorithm proposed in [13] takes roughly 6–8 s to smooth a 174,000-frame trace on an SGI workstation with a 150-MHz R4400 processor. However, it is conceivable that this overhead may be unacceptably high in an on-line situation. Large-scale servers handle hundreds of clients, and may have to deal with several on-line computations concurrently. In addition, while it is expected that servers would be optimized for I/O-intensive operations, modern web servers may have to perform compute-intensive activities and computation overheads cannot be ignored.

4 Results establishing the relationship between optimal schedules

We assume that any real-world "situation" may be modeled by a pair of constraints (D, B). Given two physically different situations $(D1, B1)$ and $(D2, B2)$ with optimal curves $A1$ and $A2$, we present results, which, given particular relationships between $D1$ and $D2$, and between $B1$ and $B2$, establish relationships between $A1$ and $A2$. In Sect. 5, we apply these results to relate schedules arising from different client buffer sizes and different worst case network jitter estimates.

We begin by presenting a fundamental result, the *Domination theorem*, which we find useful in subsequent results.

Definition 1 (Domination). *A function $f(t)$ is said to dominate a function $g(t)$ over an interval $[p, q]$ iff $f(t) \geq g(t)$, $\forall t$, $p \leq t \leq q$.*

Theorem 1 (Domination theorem). *Let $B2$ dominate $B1$ over an interval $[p, q]$, $D2$ dominate $D1$ over $[p, q]$, $A2(p) \geq A1(p)$ and $A2(q) \geq A1(q)$. Then $A2$ dominates $A1$ over $[p, q]$.*

See Appendix A for proof. Briefly, the proof is by contradiction. We assume a maximal segment $[s, u)$ such that $A1$ dominates $A2$ over $[s, u)$, $p < s \leq u \leq q$, $A2(s-1) \geq A1(s-1)$ and $A2(u) \geq A1(u)$. We then show that $A1$ has no points of rate -decrease (concave) over $[s, u)$ and $A2$ has no points of rate increase over $[s, u)$. From this, we show $A1(u) > A2(u)$ and arrive at a contradiction.

We now introduce the notion of *critical points* of an optimal schedule.

Definition 2 (critical points). *t is a critical point of an optimal schedule if the client buffer is either empty ($A(t) = D(t)$) or full ($A(t) = B(t)$) at time t. In the former case, the critical point is referred to as concave and in the latter case as convex.*

All change points of an optimal schedule are also critical points for that schedule, but the converse need not be true.[2] Given the critical points of an optimal schedule, and stored $D(t)$ values, it is easy to reconstruct the schedule itself.

In Sect. 3, we have seen that a change of client buffer size displaces the work-ahead constraint B upwards or downwards, while a change of worst case network jitter estimate displaces the constraint B to the left or right. We now present the *Refinement theorem*, that relates the critical points of schedules $A1$ and $A2$, optimal respectively for constraints $(D1, B1)$ and $(D2, B2)$, where $D1$ and $D2$ are identical, $B2$ dominates $B1$, and $B1$ is obtained either by displacing $B2$ downwards or to the right. We will use the *Refinement theorem* in Sect. 5 to relate optimal schedules arising from different client buffer sizes and different worst case network jitter estimates.

Theorem 2 (Refinement theorem). *Let $D1(t) = D2(t)$ and $B1(t) \leq B2(t)$, $\forall t, 0 \leq t \leq N$. Then,*
(a) The concave critical points of $A1$ are a superset of the concave critical points of $A2$.
(b) (i) If $B2(t) \leq B1(t) + k\ \forall t, 0 \leq t \leq N$, then, $A2(t) \leq A1(t) + k\ \forall t, 0 \leq t \leq N$. (ii) Further, if over an interval $[p, q]$, $B2(t) = B1(t) + k\ \forall t, p \leq t \leq q$, then, the convex critical points of $A1$ in $[p, q]$ are a superset of the convex critical points of $A2$ in $[p, q]$.
(c) (i) If $B2(t) \leq B1(t + j)\ \forall t, 0 \leq t \leq N - j$, then, $A2(t) \leq A1(t + j)\ \forall t, 0 \leq t \leq N - j$. (ii) Further, if over an interval $[p, q]$, $B2(t) = B1(t + j)\ \forall t, p \leq t \leq q$, then, for any convex critical point w of $A2$, $p \leq w \leq q$, $w + j$ is a convex critical point of $A1$.

Refer to Appendix A for proof. (a) is proved observing that $A2$ dominates $A1$ over $[0, N]$. To prove (b), we displace the curves $D1, B1$ upwards by k to get the curves $D1', B1'$ and then apply Theorem 1 for $(D1', B1')$ and $(D2, B2)$. (c) is proved in a similar fashion to (b), except that $(D1', B1')$ is obtained by displacing the curves $D1, B1$ to the left by j.

[2] For example, consider the case where all N frames have the same size. The optimal schedule is a single CBR segment with no change-points, yet $\forall t, t \in \{1, 2, \ldots, N\}$, t is a concave critical point.

Finally, we present the *Convergence theorem*, which we find useful in reducing on-line schedule recomputation time in the context of playback restart after an interaction.

Definition 3 (Convergence). *Schedule $A1$ is said to converge with schedule $A2$ at a time p, if $\forall t, t \geq p, A1(t) = A2(t)$.*

Theorem 3 (Convergence theorem). *Let $\exists q$ such that $B2(t) = B1(t), D2(t) = D1(t)\ \forall t, q \leq t \leq N$. Let $A2(q) \geq A1(q)$. Then, $A1$ and $A2$ converge at the first concave critical point of $A2(> q)$ or the first convex critical point of $A1(> q)$, whichever is earlier.*

Refer to Appendix A for proof. It is based on the observation that $A2$ dominates $A1$ over $[q, N]$, and if $\exists p, p \geq q$ such that $A2(p) = A1(p)$, then $A2$ and $A1$ dominate each other (and are hence identical) over $[p, N]$.

5 Handling heterogeneous client buffer sizes, variable network jitter and client interaction

We apply the results presented in the previous section to the contexts of heterogeneous client buffer, variable worst case network jitter and playback resumption after an interaction. In each case, we devise methods to reduce the on-line computation time.

5.1 Heterogeneous client buffer sizes

In this section, we apply the *Refinement theorem* to show that the critical points of the optimal schedule for any client buffer size $b1$ are a superset of the critical points of the optimal schedule for any client buffer size $b2$, $b2 > b1$. We then present an algorithm, by which the server can precompute the optimal schedules for all possible client buffer sizes in an efficient manner. Further, the server need not maintain a copy of the optimal schedule for each possible client buffer size; rather, by storing the schedule for minimum possible client buffer size alone, the server can easily retrieve the optimal schedule for any particular buffer size.

Corollary 1. *Let $A1$ and $A2$ represent the optimal schedules for a given video and client buffers $b1$ and $b2$, respectively ($b2 > b1$). Then, the critical points of $A1$ are a superset of the critical points of $A2$.*

Proof. Let $D1, B1$ represent the constraints for a client buffer size $b1$. Let $D2, B2$ represent the constraints for a client buffer size $b2$. Now, $B1(t) = \min\{D1(t-1) + b1\}$ and $B2(t) = \min\{D2(t-1) + b2, D(N)\}$. $D1(t) = D2(t)\ \forall t$. Let TH be such that $B2(t) = D2(t-1) + b2,\ t < TH$ and $B2(t) = D2(N),\ t \geq TH$. It is easily verified that $B2(t) - B1(t) \leq b2 - b1\ \forall t$ and $B2(t) - B1(t) = b2 - b1, 0 \leq t < TH$. Applying Theorem 2(a), the concave critical points of $A1$ are a superset of the concave critical points of $A2$. Applying Theorem 2(b), the convex critical points of $A1$ in $[0, TH)$ are a superset of the convex critical points of $A2$ in $[0, TH)$. Further, it may be easily verified that $A2$ has no convex critical points in $[TH, N]$.

```
**Find_Optimal_Schedule_Any_buffer**($d(t)$)
 1. Call Find_Optimal_Schedule($d(t)$, $\infty$). Denote this by $A^*_\infty$. (Salehi-96 algorithm on infinite client buffer)
 2. Initialize list $Q$ by inserting for each critical point $t$ of $A^*_\infty$ the entry $< t, \infty,$ CONCAVE $>$
 3. Find $bmin$, $A_{bmin}$, $T$. (see text for definition)
 4. Set *First*, *Second* to point respectively to first and second entries of list $Q$.
 5. **while** (Second != NULL)
      6. Let $< b_{a1}, a1, f_{a1} >$ and $< b_{a2}, a2, f_{a2} >$ be the entries pointed to by *First* and *Second*, respectively.
      7. **if** ($\exists t, a1 < t < a2$ and $t$ is a critical point of $A_{bmin}$)
           8. $< b, L >= Refine(< a1, f_{a1} >, < a2, f_{a2} >, d(t))$. (see text for explanation of Refine)
           9. Insert each item $< a3, f_{a3} >$ of list $L$ into list $Q$ as $< b, a3, f_{a3} >$
           10. Reset *Second* to point to the entry immediately to the right of *First*.
      **else** /* Interval requires no further refinement.*/
           11. Advance First and Second by one step each.
```

Fig. 4. Algorithm β. The algorithm precomputes and stores the optimal schedule for all possible client buffer sizes. $d(t)$ represents the frame size trace of the given video

Table 2. Procedure $Refine(< a1, f_{a1} >, < a2, f_{a2} >, d(t))$. A list $< b_{a3}, L >$ is returned, where b_{a3} is obtained as the maximum of an appropriate expression and L is the list of all $< a3, f_{a3} >$, such that the maximum is attained at t = $a3$. See text for details

f_{a1}	f_{a2}	b_{a3}	(f_{a3})
concave	concave	$\max_{a1<t<a2, t\in T} D(a1) - D(t-1) + \frac{[D(a2)-D(a1)]*(t-a1)}{(a2-a1)}$	(convex)
convex	convex	$\max_{a1<t<a2, t\in T} D(t) - D(a1-1) - \frac{[D(a2-1)-D(a1-1)]*(t-a1)}{(a2-a1)}$	(concave)
concave	convex	$\max_{a1<t<a2, t\in T} \begin{cases} D(a1) - D(a2-1) + \frac{(a2-a1)*[D(t)-D(a1)]}{t-a1} (concave) \\ \frac{[D(a1)-D(t-1)]*(a2-a1)}{a2-t} + \frac{[D(a2-1)-D(a1)]*(t-a1)}{a2-t} (convex) \end{cases}$	
convex	concave	$\max_{a1<t<a2, t\in T} \begin{cases} [D(a2) - D(a1-1)] - \frac{(a2-a1)*[D(t-1)-D(a1-1)]}{t-a1} (convex) \\ \frac{D(t)*(a2-a1)}{a2-t} - \frac{D(a2)*(t-a1)}{a2-t} - D(a1-1) (concave) \end{cases}$	

Let $bmin$ be the minimum possible client buffer ($bmin$ = size of largest frame of presentation). Let A_{bmin} denote the optimal schedule of transmission for a client buffer $bmin$ and T denote the set of all critical points of A_{bmin}. We find the notion of *transition buffer of a critical point* useful.

Definition 4 (Transition buffer). *Let $t \in T$. Then, the transition buffer b_t of critical point t is that buffer such that t is a critical point of the optimal schedule of transmission for a client buffer $b \leq b_t$ and not for any higher client buffer.*

We now present an algorithm β (formally in Fig. 4) to precompute the optimal schedules for all possible client buffer sizes. The algorithm constructs a list Q (which can be stored in a file and retrieved whenever needed). Each entry of list Q is a triplet of the form $< t, b_t, f_t >$, which indicates that $t \in T$, b_t is a transition buffer of t and f_t is a flag that indicates if t is concave or convex. The entries in list Q are stored in increasing order of t.

The basis of the algorithm β is the procedure *Refine*. Given that $a1, a2 \in T$ and have transition buffers b_{a1}, b_{a2}, respectively, and given there exists no $t \in T$ in $(a1, a2)$ with a transition buffer $b_t \geq \min\{b_{a1}, b_{a2}\}$, Refine finds $< a3, b_{a3}, f_{a3} >$ such that b_{a3} = $\max\{b_t | a1 < t < a2, t \in T, b_t$ is transition buffer of t$\}$ and $a3$ is the critical point for which the maximum is attained, f_{a3} indicating if $a3$ is convex or concave. In case of a tie, Refine returns a list L of all such $< a3, f_{a3} >$. Table 2 summarizes how b and list L may be identified given $a1$ and $a2$, f_{a1} and f_{a2}. (f_{a1},f_{a2} are flags that indicate whether the respective critical points are convex or concave.) A derivation of the results presented in Table 2 can be seen in Appendix B.

Algorithm β builds list Q as it progresses and operates as follows. The critical points of the optimal schedule for infinite client buffer[3] are computed directly using the optimal algorithm in [13](Step 1), and for each point, a corresponding entry is inserted in Q (Step 2). At any time, in Step 6, the following invariant holds: the pointers *First* and *Second* point to *adjacent* entries $< b_{a1}, a1, f_{a1} >$ and $< b_{a2}, a2, f_{a2} >$ of list Q, and an entry already exists in Q $\forall t,\ t < a1,\ t \in T$. Step 7 verifies if $\exists t \in T$ in the interval $(a1, a2)$. If this is the case, procedure *Refine* is applied on $< a1, f_{a1} >$ and $< a2, f_{a2} >$ in Step 8, and the critical points returned by *Refine* are inserted in Q (Step 9). The process continues until there is an entry in Q for each critical point of A_{bmin}.

Note that once Q is computed and stored by the server, the critical points of the optimal schedule for a particular client buffer b may be obtained by inspecting each entry $< t, b_t, f_t >$ of Q and choosing those entries $< t, f_t >$ such that $b_t \geq b$. With the list of critical points, the optimal

[3] Interestingly, the optimal schedule for infinite client buffer as computed according to the algorithm in [13] is identical to the schedule produced using another smoothing algorithm – the critical bandwidth algorithm [6]

Consider the following artificially constructed example to illustrate the working of algorithm β and procedure Refine.
Assume for a video, N = 6 and $d(t)$ is given by $< 1000, 500, 375, 1000, 1125, 1250 >$.
Then, $D(t)$ is given by $< 1000, 1500, 1875, 2875, 4000, 5250 >$.
Also, $bmin$ = 1250 (minimum client buffer size = size of largest frame).
Further, T = Set of critical points of optimal schedule for client buffer size $bmin$
= $\{1, 4, 5, 6\}$ (Obtained by running the Salehi-96 Algorithm).
The optimal schedule for infinite client buffer has two concave critical points: {1,6}.
We now trace the execution of the algorithm and show how Q is constructed.

At start of while loop in algorithm β, $Q =< 1, \infty, concave >, < 6, \infty, concave >$.

1. Refine is called with $< 1, concave >$ and $< 6, concave >$. Refine now computes with $a1$ = 1, $a2$ = 6,
   $\max_{a1<t<a2, t\in T} D(a1) - D(t-1) + \frac{[D(a2)-D(a1)]*(t-a1)}{(a2-a1)}$
   The maximum occurs at t = 4 and is 1675. Thus now:
   $Q =< 1, \infty, concave >, < 4, 1675, convex >, < 6, \infty, concave >$.
2. Refine is called with $< 1, concave >$ and $< 4, convex >$. But as $\nexists t, t \in T, 1 < t < 4$,
   Q is unchanged and the interval $< 1, 4 >$ needs no further refinement.
3. Refine is called with $< 4, convex >$ and $< 6, concave >$. Refine now computes with $a1$ = 4 and $a2$ = 6,
   $\max_{a1<t<a2, t\in T} \begin{cases} [D(a2) - D(a1-1)] - \frac{(a2-a1)*[D(t-1)-D(a1-1)]}{t-a1} (convex) \\ \frac{D(t)*(a2-a1)}{a2-t} - \frac{D(a2)*(t-a1)}{a2-t} - D(a1-1) (concave) \end{cases}$
   The maximum occurs at t = 5, for the first(convex) subexpression, and is 1375. Thus now,
   $Q =< 1, \infty, concave >, < 4, 1675, convex >, < 5, 1375, convex >, < 6, \infty, concave >$
4. The intervals $< 4, 5 >$ and $< 5, 6 >$ cannot be refined further; The algorithm terminates with
   $Q =< 1, \infty, concave >, < 4, 1675, convex >, < 5, 1375, convex >, < 6, \infty, concave >$

Determination of Optimal Schedule for a given client buffer b using Q:
Each triplet $< t, b_t, f_t >$ of Q is examined and those $< t, f_t >$ are chosen for which $b_t \geq b$. For example, for b = 1400, the critical points are: $< 1, concave >, < 4, convex >, < 6, concave >$. With the list of critical points and with stored $D(t)$, the optimal schedule may be reconstructed.

Fig. 5. Example to illustrate the working of algorithm β and procedure Refine.

Table 3. K is much less than N (see text). $bmin$ is fixed to be the size of the largest frame for each trace

Trace	$bmin$(KB)	N	K
Advertisements	10.320	16316	970
Jurassic Park	14.954	40000	1180
MTV	31.426	40000	1114
Star Wars	23.158	174054	3832
Wizard of Oz	42.891	41760	1139

schedule for a client buffer b may be easily reconstructed. Figure 5 gives a complete example that illustrates the working of algorithm β and procedure Refine.

Let K be the number of critical points in the optimal schedule for minimum possible client buffer ($K = |T|$). Theoretically, K maybe N, (the number of frames in the video). However, we find on experimentation with several VBR-compressed MPEG-1 video frame-size traces[4] that K is in practice much smaller than N. This fact is illustrated in Table 3. It is easily verified that Algorithm β has a $\max\{O(K^2), O(N)\}$ complexity. Storage requirement is $O(K)$ and the time of retrieval of the optimal schedule for a given client buffer takes $O(K)$ time.

5.2 Variable worst case network jitter

In this section, we apply the *Refinement theorem* to show an important relationship between the critical points of the optimal schedules for two different worst case network jitter estimates. We then show that it is possible for the server to store the optimal schedule for maximum possible worst case network jitter alone, and yet easily retrieve the optimal schedule for any particular worst case network jitter. A lot of the discussion in this section parallels the discussion concerning heterogeneous client buffer sizes in Sect. 5.1.

Corollary 2. *Let $A1$,$A2$ represent the optimal schedule for worst case network jitter $J1$ and $J2$ respectively, $J1 < J2$. Then, (a) the concave critical points of $A2$ are a superset of the concave critical points of $A1$. (b) if $A1$ has a convex critical point at t, $A2$ has a convex critical point at $t + J2 - J1$.*

The proof is similar to Corollary 1 and is omitted.

Assume that, during the presentation, the worst case network jitter estimate is bounded by J_{max}, and that the optimal schedule for the estimate J_{max} has m critical points. Then, it is possible to precompute and store the optimal schedules for all possible estimates of worst case network jitter $j, j \leq J_{max}$ using $O(m)$ space. This can be done by storing triplets of the form $< t, J_t, f_t >$, where, f_t is a flag that indicates concave or convex. If f_t is concave, then t is a concave critical point in the optimal schedule for worst case network jitter j, $j \geq J_t$, and if f_t is convex, then $t+j-J_t$ is a convex critical point in the optimal schedule for any worst case network jitter j, $j \geq J_t$.

Finally, we believe that it is possible to develop an efficient algorithm to actually compute and store triplets in the above fashion. We defer development of such an algorithm to future work.

[4] All traces were VBR-encoded using an MPEG-1 software encoder. The videos had length as follows (in minutes): *Advertisements* [8] 9, *Jurassic* [12] 28, *MTV* [12] 28, and *Wizard* [9] 23. *Star Wars* [7] was about 2 h long.

5.3 Playback restart after an interaction

In this section, we apply the Convergence theorem to the context of client playback after an interaction, and evaluate its impact on reduction in schedule recomputation time.

For a given video of N frames and a given client buffer size b, let D,B represent the work-ahead constraints, and A the optimal schedule for the entire video. Let a client index randomly to a frame f, flushing out its buffer. Let $A1$ represent the new optimal schedule of transfer for frames $f+1$ to N. Then, by a direct application of the Convergence theorem, it follows that A and $A1$ converge by the first convex change point of $A1$ or the first concave change point of A ($> f$), whichever is earlier. While convergence occurs actually at the first convex critical point of $A1$ or the first concave critical point of A, we find dealing with change - points more convenient algorithmically, and hence we use a slightly weaker version of the Convergence theorem.

The on-line recomputation of $A1$ proceeds in identical fashion to the optimal algorithm, except that, whenever a change point, say c is identified, it is verified if c is a change point of A as well, in which case further computation of $A1$ is stopped. Beyond c, $A1$ and A are identical.

We now empirically evaluate the impact of the Convergence lemma in reducing on-line schedule recomputation time, by experimentation with several MPEG frame-size traces. We assume that the client may randomly index to any frame i ($0 \leq i < N$) with equal probability. Let $CL(i)$, $BR(i)$ denote respectively the number of frames to be inspected in an on-line recomputation of the optimal schedule for frames $i+1$ to N, if the Convergence theorem were used, and if a blind recomputation were done. (Note that $BR(i) = N - i$). Let $w(i) = \frac{CL(i)}{BR(i)}$. Let $w_{avg} = \frac{\sum_{i=0}^{N-1} w(i)}{N}$, $CL_{avg} = \frac{\sum_{i=0}^{N-1} CL(i)}{N}$. w_i, w_{avg} are measures of the impact of the Convergence theorem in reducing on-line computation overhead, while $CL(i), CL_{avg}$ are measures of the absolute computation overheads involved when the Convergence theorem is used.

Figure 6 shows the variation of w_{avg} with client buffer size for several MPEG-1 traces. For each trace, note that w_{avg} increases with client buffer size. This is due to the fact that, when the segment lengths of the original schedule are relatively small, the change point of convergence is likely to occur faster. (Also, the Convergence theorem becomes more significant as the length of the trace increases; note that the *Star Wars* trace is four times as long as other traces.) From the figure, we see that for most traces and client buffer sizes $\leq 512KB$, there is an average reduction of more than five times ($w_{avg} \leq 0.2$) in the number of frames to be inspected when the *Convergence theorem* is used.

Figure 7 shows the variation of CL_{avg} with the client buffer size for several traces. On average, an on-line recomputation inspects less than 5000 frames across all traces for buffer sizes $\leq 512KB$ when the Convergence theorem is used.

In conclusion, the Convergence theorem can have appreciable impact in reducing recomputation time of a new schedule. Even in cases where the Convergence theorem may not have a significant influence, we still recommend its use in case of an on-line recomputation as there is no "penalty" involved.

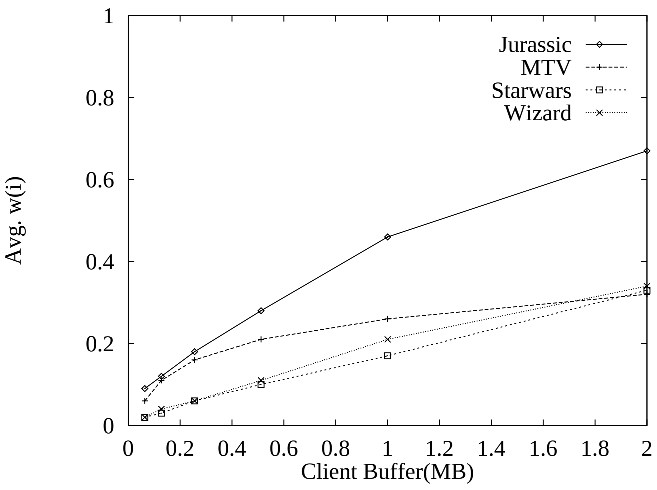


Fig. 6. w_{avg} vs client buffer size

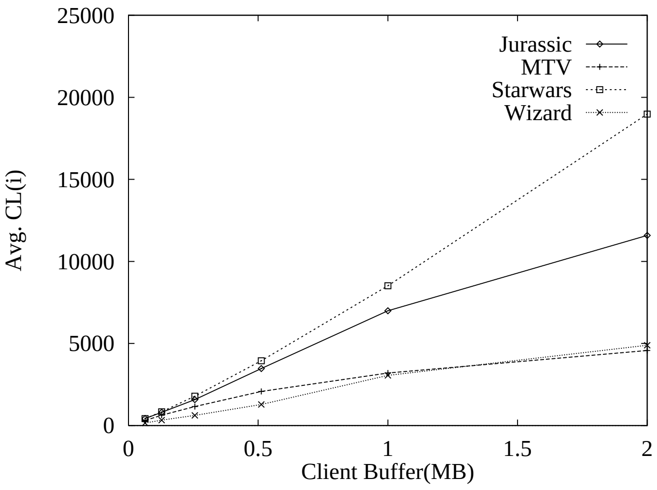


Fig. 7. CL_{avg} vs client buffer size

6 Approximation scheme

In this section, we propose an approximation scheme that works by altering the work-ahead constraints and performing optimal smoothing on the altered constraints. The resulting schedule is feasible and close to being optimally smooth with respect to the original constraints, yet it takes significantly less time to compute.

The optimal algorithm proceeds by inspecting every frame as a possible candidate for being a change point. Yet it may be verified that a frame with size 0 cannot be a change point and the optimal algorithm can skip inspecting such a frame. The approximation scheme readjusts the frame size sequence in such a manner that many frames have a size 0. The video is assumed to consist of N/g blocks of g consecutive frames each (assume for simplicity that N is a multiple of g). The sequence is now readjusted so that, in each block, the entire data of all the frames of the block is attributed to its first frame, all other frames having size 0. Formally, if f_j^i and F_j^i respectively denote the size of

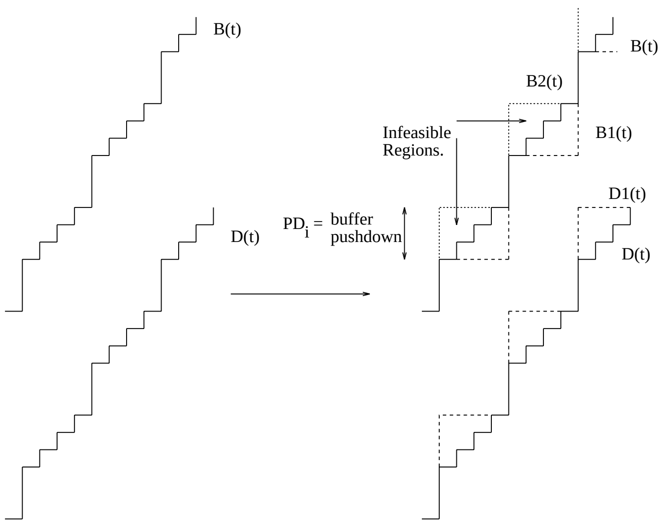


Fig. 8. Illustration of the approximation scheme

frame j of block i in the original and new sequences, then, $F_1^i = \sum_{j=1}^{g} f_j^i$ and $F_j^i = 0, j \neq 1$. Figure 8 shows how the original constraints $D(t)$,$B(t)$ may be modified to give the new constraints $D1(t)$ and $B2(t)$ after readjustment of the sequence. Note, however, that the schedule arising from smoothing with constraints $D1(t), B2(t)$ may be infeasible. We tackle this by adjusting the curve $B2(t)$ downwards (thus we do not make fullest possible use of the client buffer). Let $PD_i = \sum_{j=2}^{g} f_j^i, 1 \leq i \leq N/g$. Then, the altered constraint $B1(t)$ is obtained as $B1(t) = B2(t) - PD_{\lceil \frac{t-1}{g} \rceil}, t > 1, B1(1) = B2(1)$. Smoothing is now performed with the work-ahead constraints $D1(t), B1(t)$ and we refer to the resulting schedule as the approximate schedule. Only N/g frames have to be examined as all other frames have a size 0; however, the schedule is not optimal, as we start off with a burstier sequence and do not make fullest possible use of the client buffer.

We now investigate the performance of the approximation scheme by experimentation with several MPEG frame size traces. We demonstrate that the smoothness of the optimal and approximate schedules are comparable. Further, we expect that the approximation scheme achieves a reduction in computation time close to the block size g (as it examines only about $1/g$ of the frames that the optimal scheme does) and we show that this is indeed the case.[5]

We first present some of the parameters that we use in our experiments. Typically, the very first frame of the video is much larger than its immediately subsequent frames and a start-up latency of s units may be introduced to smooth its transmission [13]. However, the approximation scheme results in an increase in the size of the first frame due to aggregation, which offsets the benefits of the start-up latency. Further, with the approximation scheme, no data is transmitted in the last $g - 1$ (g is the block size) frame times. To avoid these problems, we do not apply the approximation scheme to the first $L1$ and the last $L2$ blocks of the video,

[5] The smoothing algorithm runs in $O(N)$ time, where N is the number of frames examined. In our experiments, we actually used the $O(N^2)$ version of the smoothing algorithm; however, this version too runs in close to linear time in practice.

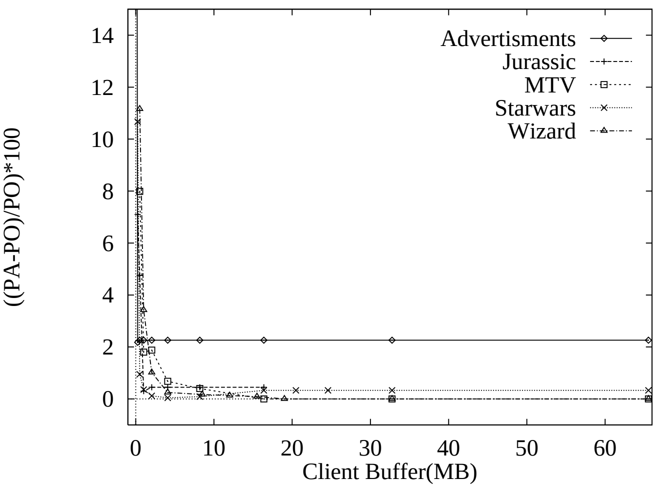


Fig. 9. The % increase in peak using the approximation scheme

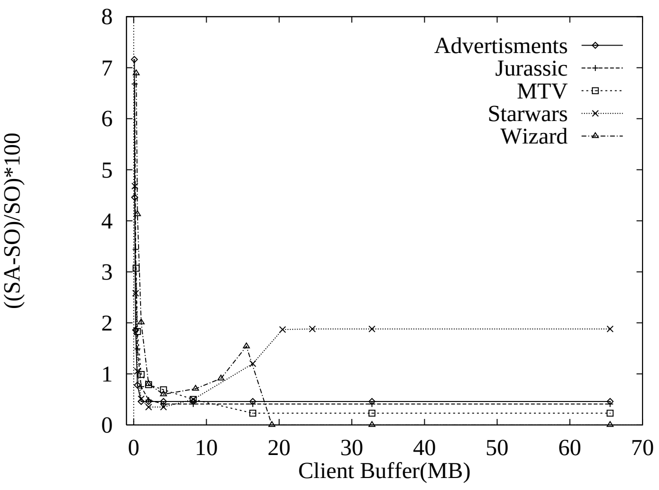


Fig. 10. The % increase in standard deviation using the approximation scheme

and instead we retain the original frame size sequence for these blocks. In the experiments described below, we fix $s = 15$, $L1 = 15$ and $L2 = 15$. For each video, we fixed the block-size g to be equal to the GOP of that video. The GOP of a video is the number of frames between consecutive I-pictures in its compression pattern [1]. The GOP of *Star Wars* is 12, of *Advertisements* 6, of *Jurassic* 12, of *MTV* 12, and of *Wizard* 15.

Let PA,SA and CA (PO,SO,CO) denote respectively the peak, standard deviation and computation time of the approximate (optimal) schedule. Figures 9 and 10 plot the % increase (as compared to optimal) in peak and standard deviation when the approximation scheme is used against client buffer size for several traces. Across all traces, for client buffers greater than 1 MB, the increase of peak is less than 2%. For most traces, and client buffer sizes greater than 4 MB, the increase in peak is less than 0.5%. The increase in standard deviation is less than 2% for buffer sizes greater than 1 MB for most traces. For small buffer sizes ($< 1MB$), however, the peak of the approximate schedule may grow to as large as 60% higher than optimal (not shown in graph). This is understandable, because for small buffer sizes, the cost of under-utilizing client buffer becomes significant.

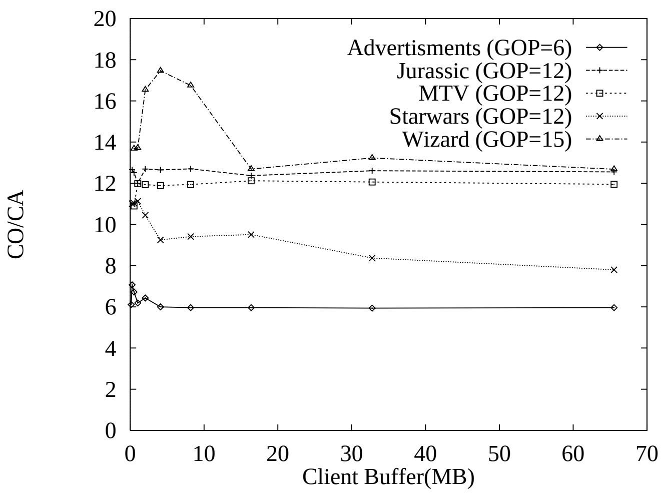


Fig. 11. Reduction in computation time (CO/CA) using the approximation scheme. The block size used for each trace was fixed at the GOP of the trace. The GOP of the traces are as follows: Wizard 15, Advertisements 6, Star Wars 12, Jurassic Park 12 and MTV 12. For each video, a speed-up close to block size was achieved across all client buffer sizes

Figure 11 shows the speed-up obtained on using the approximate algorithm as compared to the optimal one, as a function of client buffer. The speed-up is measured on a Pentium 266-MHz processor.[6] Only schedule computation time is considered, and time for input, preprocessing and output is not. As expected, the speedup is close to the block size used for each trace and is not significantly affected by the client buffer size. For example, *Wizard* (block size 15) shows a 13–17-fold speed-up, and Jurassic (block size 12) shows around a 12-fold speed-up across all client buffer sizes.

Finally, in [14], we investigate the impact of varying the block size on the performance of the approximation scheme. Our findings indicate that (i) the speed-up achieved is proportional to the block size, and (ii) in general, larger block-sizes result in poorer approximations; however, exceptions do occur for MPEG video because of the encoding pattern. See [14] for details.

7 Related work

Feng et al. [4] have considered the problem of interactivity in the context of work-ahead smoothing. They introduce the notion of "VCR window" and demonstrate that, for large buffers above 25 MB, most requests for rewind could be handled within the window itself, from the data residing within the smoothing buffer. In such a case, no recomputation of the schedule of transfer is required. On a rewind, transmission by the computed schedule of transfer is stopped and is resumed when normal playback reaches the original point of play. However, it is not clear that this approach would work well in the case of smaller client buffers. Further, their method is not applicable when there is a change in smoothing constraints during the presentation (such as dynamic change of client buffer or worst case network jitter estimates) when a recomputation of the optimal schedule becomes unavoidable.

Dey et al. [2] have considered the problem of playback restart after an interaction. They propose an algorithm by which the restart latency can be minimized, assuming transfer bounded by a particular peak rate determined by the original schedule of transfer. They do not recompute a new schedule of transfer on-line, rather they transmit at the peak rate until convergence with the old schedule of transfer occurs. While this approach has advantages in that the existing network resources are sufficient and no fresh reservation needs to be made, the disadvantage is that there is no direct control over the resulting start-up latency. Again, their technique is not applicable when there is a change of smoothing constraints during the presentation.

Rexford et al. [11] have considered work-ahead smoothing of *live* video, where there is only a limited knowledge of frame sizes in the future (made possible by the introduction of delay at the source). The authors examine schemes in which the optimal smoothing algorithm is periodically executed on-line to compute a transmission schedule over the next window of frames. The performance of these schemes is compared to that of optimal off-line transmission. An interesting question that has not been explored is the cumulative costs associated with repeated on-line computation.

8 Summary and conclusions

Work-ahead smoothing is a technique whereby a server transmits stored video to a client in accordance with a computed schedule of transfer. Past work [13] established an algorithm that computes a schedule, which optimally minimizes peak, variance and rate variability of the transmitted stream, under the assumptions of fixed client buffer size, known worst case network jitter and strict playback of the client video. In this paper, we have considered practical situations characterized by heterogeneous client buffer sizes, dynamically changing worst case network jitter estimates, and client interactions such as fast-forward and rewind. Such a setting requires an on-line computation of the schedule of transfer. We have presented the *Domination theorem*, *Refinement theorem* and the *Convergence theorem* which establish important relationships between optimal schedules obtained under different sets of constraints. We then use these results to devise methods for reducing on-line computation time in various practical situations. We summarize our results below.

(i) Based on the Refinement theorem, we have proposed an $O(K^2)$ algorithm that can precompute the optimal schedules for all possible client buffer sizes and store them in $O(K)$ space, such that retrieval of the schedule for a particular buffer size can be done in $O(K)$ time. Here, K is the number of critical points in the optimal schedule for smallest possible client buffer size, and experiments reveal that K is significantly smaller than N, the number of frames in the video.

(ii) It is theoretically possible to precompute and store the optimal schedules of transfer for all possible estimates of worst case network jitter $j, j < J_{max}$ in $O(m)$ space. Here,

[6] The speedup is measured using UNIX system calls getitimer() and setitimer(), using the flag ITIMER_PROF, so that only CPU-time for the process is considered.

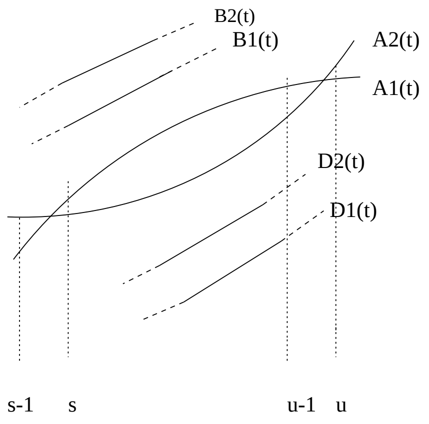


Fig. 12. Illustration for Theorem 1. All functions are step functions shown as curves for convenience

m is the number of critical points in the optimal schedule for worst case network jitter J_{max}.

(iii) In the context of on-line recomputation of the optimal schedule on playback resumption after an interaction, the *Convergence theorem* shows at what point in time the recomputed and original schedules converge. The theorem has a significant impact in reducing on-line computation time for small client buffer sizes and long videos. For client buffer sizes $\leq 512KB$, the number of frames to be examined in an on-line recomputation is reduced more than fivefold on average.

(iv) We have proposed an approximation scheme that works by altering the work-ahead constraints and performing smoothing on the altered constraints. Experimental evaluation shows that, for client buffer sizes $> 1MB$, the approximate schedule has peak and standard deviation within 2% of optimal, yet it may be computed as much as 10–15 times faster.

Appendix

A Proofs of theorems

Proof of Theorem 1

We will consistently use the notation presented in Table 1 using the suffixes 1,2 to distinguish between the two situations. The proof is by contradiction. Let s be the smallest time at which $A2(s) < A1(s), p < s < q$. Let $u = \min\{t|t > s, A2(t) \geq A1(t)\}$. Clearly, u must exist, as $A2(q) \geq A1(q)$. (Refer Fig. 12)

Claim 1. *$a1(t)$ is non-decreasing, $a2(t)$ non-increasing over [s,u], and $a1(s) > a2(s)$.*

Proof. $\forall t, s \leq t < u, A1(t)$ (strictly) $> A2(t) \geq D2(t) \geq D1(t)$. Hence, there is no concave change point of $A1$ in [s,u), for if t were concave, $A1(t) = D1(t)$. Hence, $\forall t, s \leq t < u, a1(t+1) \geq a1(t)$.
Similarly, $\forall t, s \leq t < u, A2(t) < A1(t) \leq B1(t) \leq B2(t)$ and $a2(t)$ is non-increasing over [s,u].
$a1(s) > a2(s)$ is obvious. $(A1(s) > A2(s), A1(s-1) \leq A2(s-1))$.

From Claim 1, we have, $a1(u) \geq a1(s) > a2(s) \geq a2(u)$. But, $a1(u) < a2(u)$ (as $A1(u) \leq A2(u)$, $A1(u-1) > A2(u-1)$), and we have a contradiction.

We find the following fact useful in our later proofs.

Fact 1. *Let A ($A1$) represent the optimal curve of transmission for the constraints D ($D1$) and $B(B1)$. Let $D1$ and $B1$ both be displaced upwards (downwards,right,left) from D and B by the same constant k. Then, $A1$ is also displaced upwards (downwards, right,left) from A by the same constant k.*

Proof of Theorem 2

(a) $A2(0) = D2(0) = D1(0) = A1(0)$. Similarly, $A2(N) = A1(N)$. Applying Theorem 1, $A2$ dominates $A1$ over $[0, N]$. Let w be a concave critical point of $A2$. Then, $A2(w) = D2(w)$. Also, $D2(w) = D1(w)$ (given) and $A2(w) \geq A1(w)$ ($A2$ dominates $A1$). From the above, $D1(w) \geq A1(w)$. But for feasibility of $A1$, $A1(w) \geq D1(w)$. Hence, $A1(w) = D1(w)$ and w is a concave critical point of $A1$.
(b)(i) Construct $B1'$, $D1'$ by displacing $B1$,$D1$ upwards by k. That is, $B1'(t) = B1(t) + k, D1'(t) = D1(t) + k \ \forall t \ 0 \leq t \leq N$. By Fact 1, the optimal schedule for $(D1', B1')$, say $A1'$, is displaced upwards from $A1$ by k, that is, $A1'(t) = A1(t) + k \ \forall t, 0 \leq t \leq N$. It is easily verified that $B1'$ dominates $B2$ over $[0, N]$, $D1'$ dominates $D2$ over $[0, N]$. Further, $A1'(0) > A2(0)$, $(A1'(0) = A1(0) + k = D1(0) + k = D2(0) + k = A2(0) + k)$ and similarly, $A1'(N) > A2(N)$. Hence, applying Theorem 1, $A1'$ dominates $A2$ over $[0, N]$. That is, $\forall t \ 0 \leq t \leq N, A2(t) \leq A1'(t) = A1(t) + k$.
(ii) Let w, $p \leq w \leq q$ be a convex critical point of $A2$. Then, $A2(w) = B2(w) = B1(w) + k$. From (i), $A2(w) \leq A1(w) + k$. Hence, we have, $B1(w) \leq A1(w)$. But for feasibility of $A1$, $A1(w) \leq B1(w)$. Hence, $A1(w) = B1(w)$ and w is a convex critical point of $A1$.
(c) The proof is similar to (b), except that $B1', D1'$ are constructed by displacing $B1, D1$ left by j.

Proof of Theorem 3

$B2$ dominates $B1$ over $[q, N]$, and $D2$ dominates $D1$ over $[q, N]$. $A2(q) \geq A1(q)$. Also, $A2(N) = A1(N)$. $(A2(N) = D2(N) = D1(N) = A1(N))$. Hence, $A2$ dominates $A1$ over $[q, N]$. Let u be the first concave critical point of $A2$ $(u > q)$. We have, $A1(u) \leq A2(u) = D2(u) = D1(u)$. But, we have $A1(u) \geq D1(u)$ for feasibility of $A1$, and hence we have $A1(u) = D1(u) = A2(u)$. If v is the first convex critical point of $A1$, $v > q$, then, similarly, we can show $A1(v) = A2(v)(= B1(v) = B2(v))$. Let $r = \min\{u, v\}$. Then, $A1(r) = A2(r)$. Now, applying Theorem 1 twice over $[r, N]$, $A1$ and $A2$ dominate each other over $[r, N]$. This is possible only if $A1(t) = A2(t) \ \forall t, \ t \geq r$.
Next, it is easy to see that convergence does not occur before r. If convergence occurs at time p, then, $A1(p-1) < A2(p-1)$, $A1(p) = A2(p)$ and $A1(p+1) = A2(p+1)$. Hence, p is either a point of rate decrease (concave change point) of $A1$ or a point of rate increase (convex change point) of $A2$. In the former case, $A2(p) = A1(p) = D1(p) = D2(p)$ or p is a

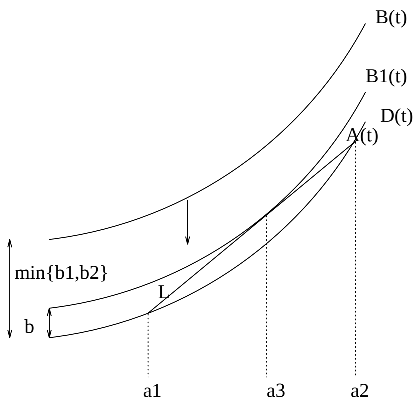


Fig. 13. Illustration for proof of Observation 8

concave critical point of $A2$. Similarly, in the latter case, it may be verified p is a convex critical point of $A1$.

B Derivation of Table 2

Table 2 is derived considering each of four separate cases, where $f1$ may be concave or convex and $f2$ may be concave or convex, ($f1$ and $f2$ are as discussed in Sect. 5.1). We consider only the case where $f1$ and $f2$ are both concave. The other cases may be handled similarly. We assume close familiarity with the terminology stated in Sect. 5.1.

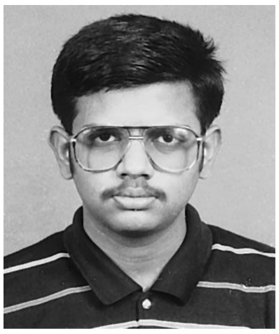
Observation 1. Let $a1$ and $a2$ be concave critical points of A_{bmin}, with transition buffers $b1$ and $b2$, respectively. Let $\forall t, a1 < t < a2$ and $t \in T$, the transition buffer for t be $< \min\{b1, b2\}$. Let $b = \max_{a1<t<a2, t\in T} D(a1) - D(t-1) + \frac{(D(a2)-D(a1))*(t-a1)}{(a2-a1)}$, with the maximum attained at $t = a3$ ($a3$ may not be unique). Then, $a3$ is a convex critical point of A_{bmin}, has a transition buffer b, and $\forall t, a1 < t < a2$ and $t \in T$, the transition buffer for t is $\leq b$.

Proof. Let $b0 = \min\{b1, b2\}$. Let $D(t)$, $B(t)$ represent the constraints for a client buffer $b0$, and $A(t)$ the optimal schedule for that buffer size. As by assumption, there are no critical points in $(a1, a2)$ for client buffer $b0$, A (refer Fig. 13) is a straight line over $(a1, a2)$. The effect of reducing the client buffer below the $b0$ translates mathematically to displacing the curve $B(t)$ downwards. By Corollary 1, $a1$ and $a2$ continue to be concave critical points of the optimal schedule as the buffer decreases. Further, as long as displacing $B(t)$ downwards does not render the straight line transmission over $(a1, a2)$ as in A infeasible, no new critical point is introduced in $(a1, a2)$. An additional convex critical point is created at that client buffer b where $B(t)$ just touches the curve $A(t)$ at some time say $a3, a1 < a3 < a2$. By Corollary 1, $a3$ is now a convex critical point in the optimal schedule calculated for any smaller buffer $\leq b$, that is the buffer b is the transition buffer of $a3$. Further, $\forall t$ such that $a1 < t < a2$ and $t \in T$, t has a transition buffer $\leq b$. It now remains to calculate b. Let $B1(t)$, $D(t)$ represent the constraints for buffer b and $A1(t)$ the optimal curve. $A1$ is identical to A over the interval $(a1, a2)$.
For $a1 < t < a2$, we have $B1(t) - A1(t) = B1(t) - A(t)$
$= [D(t-1) + b] - [A(a1) + \frac{(t-a1)*(A(a2)-A(a1))}{(a2-a1)}]$
$= [D(t-1) + b] - [D(a1) + \frac{(t-a1)*(D(a2)-D(a1))}{(a2-a1)}]$
(as $a1$, $a2$ are concave critical points of $A1$).
$= b - H(t)$, where,
$H(t) = [D(a1) - D(t-1)] + \frac{(t-a1)*(D(a2)-D(a1))}{(a2-a1)}$.
But, $B1(t) - A1(t) \geq 0 \quad \forall t, a1 < t < a2, t \in T$, and further, $B1(a3) - A1(a3) = 0$. Hence, we have, $b \geq H(t), \forall t, a1 < t < a2, t \in T$ and $b = H(a3)$. Hence, $b = \max_{a1<t<a2, t\in T} H(t)$.

Acknowledgements. We are grateful to J.D. Salehi for his constant encouragement throughout this work, and for innumerable suggestions and critical comments on the content and presentation of drafts of this paper. We thank Jayanta Dey for his encouragement and the anonymous referees of the paper for relevant remarks. The traces we used were obtained from Mark Garrett [7], Ed Knightly [8], Marwan Krunz [9] and Oliver Rose [12].

References

1. Le Gall D (1991) MPEG: A video compression standard for multimedia applications. Commun ACM 34: 46–58
2. Dey JK, Sen S, Kurose JF, Towsley D, Salehi JD (1997) Playback restart in interactive streaming video applications. In: Proc. IEEE International Conference on Multimedia Computing and Systems, June 1997, Ottawa, Canada, pp 458–465
3. Feng W, Jahanian F, Sechrest S (1997) An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video. Multimedia Syst 5(5): 297–309
4. Feng W, Jahanian F, Sechrest S (1996) Providing VCR functionality in a Constant Quality video-on-demand transportation service. In: Proc. IEEE International Conference on Multimedia Computing and Systems, June 1996, Hiroshima, Japan, pp 127–135
5. Feng W, Rexford J (1997) A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video. In: Proc. IEEE INFOCOM, April 1997, Kobe, Japan, pp 58–66
6. Feng W, Sechrest S (1995) Critical bandwidth allocation for delivery of compressed video. Comput Commun 18(10): 709–717
7. Garrett M, Willinger W (1994) Analysis, Modelling and Generation of Self-Similar VBR Video Traffic. In: Proc ACM SIGCOMM, August 1994, London, England, 24(4): 269–280
8. Knightly EW, Wrege DE, Liebeherr J, Zhang H (1995) Fundamental Limits and Tradeoffs of Providing Deterministic Guarantees to VBR Video Traffic. In: Proc ACM SIGMETRICS, May 1995, Ottawa, Canada, 23(1): 98–107
9. Krunz M, Hughes H (1995) A Traffic Model for MPEG-Coded VBR Streams. In: Proc ACM SIGMETRICS, May 1995, Ottawa, Canada, 23(1): 47–55
10. McManus JM, Ross KW (1996) Video on Demand over ATM: Constant Rate Transmission and Transport. IEEE J Select Areas Commun 14(6): 1087–1098
11. 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 IEEE NOSSDAV, May 1997, St. Louis, Missouri, pp 249–257
12. Rose O (1995) Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems. In: Proc of the 20th Annual Conference on Local Computer Networks, October 1995, Minneapolis, Minnesota, pp 397–406
13. Salehi JD, Zhang Z-L, Kurose JF, Towsley D (1996) Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. In: Proc ACM SIGMETRICS, May 1996, Philadelphia, Pennsylvania, 24(1): 222–231. An extended version appears in IEEE/ACM Trans on Networking 6(4): 397–410, 1998
14. Sanjay G (1997) Work-ahead Smoothing of Video Traffic for Interactive Multimedia Applications. Bachelor's Thesis. Indian Institute of Technology, Madras, India



SANJAY G. RAO received a B.Tech degree in Computer Science and Engineering from the Indian Institute of Technology, Madras, in 1997. He is currently a doctoral candidate at the School of Computer Science, Carnegie Mellon University. His current interests lie in research related to wide-area networks, including multicast routing and support for integrated services.

Dr. S.V. RAGHAVAN is a Professor and the Chair of the Department of Computer Science and Engineering at the Indian Institute of Technology, Madras. For the last 20 years, Dr. Raghavan has been actively involved in research and development related to performance evaluation, networks, multimedia, and protocol engineering. He was one of the founding members of the Ernet (Education and Research in Computer Networking) in India, a joint initiative of Government of India and United Nations Development Program. His current projects include development of a Multimedia Integrated Network Environment (MINE), encompassing issues arising from mobile computing, high-speed networks, and operating systems and protocol support for handling multimedia information. Dr. Raghavan has served as Chair for numerous conferences and committees, and on the editorial boards of Computer Communication Journal of IETE. He is a Fellow of IETE and a Governor of ICCC.