OPTIMAL MULTICAST SMOOTHING OF STREAMING VIDEO OVER AN INTERNETWORK

S. SEN, D. TOWSLEY, Z. ZHANG,
and J. DEY

CMPSCI Technical Report 98-77

July 1998

Optimal Multicast Smoothing of Streaming Video over an Internetwork *

Subhabrata Sen[1], Don Towsley[1], Zhi-Li Zhang[2], and Jayanta K. Dey[3]

[1] Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{sen,towsley}@cs.umass.edu

[2] Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55455
zhzhang@cs.umn.edu

[3] Interactive Multimedia Services
GTE Laboratories
Waltham, MA 02454
dey@gte.com

UMASS TECHNICAL REPORT 98 – 77

Abstract

A number of applications such as internet video broadcasts, corporate telecasts, distance learning etc. require transmission of streaming video to multiple simultaneous users across an internetwork. The high bandwidth requirements coupled with the multi-timescale burstiness of compressed video make it a challenging problem to provision network resources for transmitting streaming multimedia. For such applications to become affordable and ubiquitous, it is necessary to develop scalable techniques which can *efficiently* deliver streaming video to multiple heterogeneous clients across a heterogeneous internetwork. In this paper, we propose using *multicasting of smoothed video* and *differential caching* of the video at intermediate nodes in the distribution tree, as techniques for reducing the network bandwidth requirements of such dissemination. We formulate the multicast smoothing problem, and develop an algorithm for computing the set of *optimally* smoothed transmission schedules for the tree (such that the transmission schedule along each link in the tree has the lowest peak rate and rate variability for any feasible transmission schedule for that link) given a buffer allocation to the different nodes in the tree. We also develop an algorithm to compute the minimum total buffer allocation to the entire tree and the corresponding allocation to each node, such that feasible transmission is possible to all the clients, when the tree has heterogeneous rate constraints. MPEG-2 trace-driven performance evaluations indicate that there are substantial benefits from multicast smoothing and differential caching. For example, our optimal multicast smoothing can reduce the total transmission bandwidth requirements in the distribution tree by more than a factor of 3 as compared to multicasting the unsmoothed stream.

*The work at the University of Massachusetts was supported in part under National Science Foundation grants NCR-9523807, NCR-9508274 and CDA-9502639. The work of Jayanta K. Dey was performed when he was at the University of Massachusetts. Zhi-Li Zhang was supported by a University of Minnesota Graduate School Grant-in-Aid grant and by the NSF CAREER Award Grant No. NCR-9734428. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

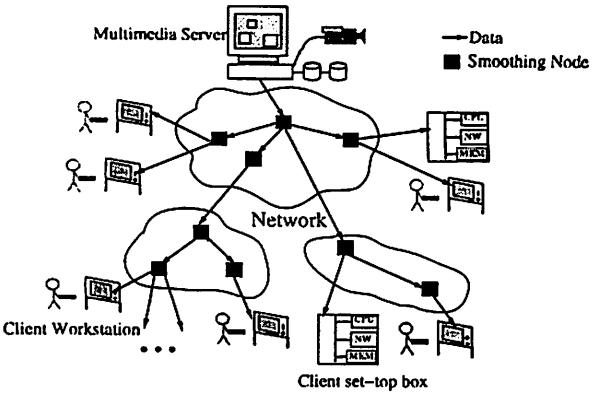


Figure 1: **Multicast Smoothing in an internetwork:** A video stream originates at a multimedia server, and travels through the internetwork, to multiple clients, including workstations and set-top boxes. Multicast smoothing service is performed at smoothing nodes within the network.

1 Introduction

A pervasive high speed internetworking infrastructure and a mature digital video technology has led to the emergence of several networked multimedia applications which include streaming video as a media component. Such applications include streaming video broadcasts, distance learning, corporate telecasts, narrowcasts, etc. Digital video traffic typically exhibits high bandwidth requirements and significant burstiness at multiple time scales, owing to the encoding schemes and the content variation between and within video scenes. When combined with other media such as audio, text, html and images, multimedia transmissions can become even burstier. The high bandwidth requirements coupled with the bursty variable-rate nature of these streams [1–3] complicates the design of efficient storage, retrieval and transport mechanisms for such media.

A technique known as *workahead smoothing* [4–6] can yield significant reductions in peak rate and rate variability of video transmissions on the end-to-end network delivery path from a server to a single client (*i.e., unicast*) over an internetwork. In smoothing, by transmitting frames early, the sender (or a smoothing node) can coordinate access with the client (or the next node downstream) and can send large video frames at a slower rate without disrupting continuous playback at the client. The frames transmitted ahead of time are temporarily stored in buffers present in the server, the client, and any intermediate network nodes. Smoothing can substantially reduce resource requirements under a variety of network service models such as peak-rate based reservation, and bandwidth renegotiation [4]. A key characteristic of the smoothed transmission schedule is that the smoothing benefit is a non-decreasing function of buffer sizes present on the end-to-end video delivery path.

For many of the applications listed above, streaming video transmission occurs from a server simultaneously to a large number of heterogeneous clients, that have different resource capacities (e.g. buffer), and scattered geographically over a heterogeneous internetwork, that has different resource capacities in different segments (see Figure 1). In this setting, an important question is how to reduce the bandwidth overhead for

such simulcast services. A naive application of unicast smoothing to a distribution tree would only consider the most constrained client buffer or most constrained link bandwidth for computing the smoothed transmission schedule to every client in the tree. This approach avoids handling the heterogeneity in the system and cannot take advantage of the presence of additional resources on other paths in the tree.

In this paper, we present a novel technique that integrates *workahead smoothing* with multicasting to efficiently transmit streamed video from a single server to several (heterogeneous) clients using a distribution tree topology as shown in Figure 1. We present and describe *differential caching*, a technique for temporal caching of video frames at intermediate nodes of the distribution tree. For prerecorded video streaming, we develop a theory, integrating smoothing with differential caching, to compute a set of optimal transmission schedules for a distribution tree, that minimizes the peak rate and variability of the video transmission along each link. When buffering at the internal nodes is the constraining resource, we present computation techniques to check whether there exists a set of feasible optimal multicast schedules to transmit a particular video. We then develop an algorithm that computes the set of optimal transmission schedules when a feasible set of buffers exists. When the link bandwidths in the distribution tree are the constraining resource, we present an algorithm that computes the minimal total buffer allocation for all the nodes in the tree, and the corresponding allocation at each node, such that there exists a set of feasible transmission schedules to distribute a particular video. MPEG trace-driven simulation results indicate that multicasting smoothed video using schedules computed by our optimal smoothing algorithm can reduce the total transmission bandwidth requirements by more than a factor of 3 as compared to multicasting the unsmoothed stream.

This paper is organized as follows. Section 2 provides a general problem overview and describes related work. Section 3 provides a brief overview on smoothing prerecorded video in a unicast setting. Section 4 presents the formal distribution tree model and problem formulations, and Sections 5 and 6 develop the solutions to the multicast smoothing problem. Section 7 evaluates our optimal multicast smoothing algorithm and Section 8 concludes the paper.

2 General Problem Description

Figure 1 illustrates a server multicasting streaming video to a number of heterogeneous clients which are connected through a heterogeneous internetwork. Part of the network constitutes a virtual distribution tree over which the video frames are multicast from the sender to the clients. The clients are leaf nodes of the distribution tree. Each internal node receives video frame data from its parents and transmits them to each of its children. The node also performs other functions described later.

The video distribution tree could be realized in a number of ways. In an active network [7], the internal nodes can be switches or routers in the network. Alternatively, analogous to the *active services* [8] approach, the nodes can be video gateway (or proxy) servers [9, 10] located at various points in the network, that perform special application-specific functions. A recent trend in the Internet world has been the growing deployment

of proxies by the ISPs for caching Web documents. It is conceivable that the nodes in a video distribution tree would be co-located at some of these Web proxy servers. In a cable network setting, the head-end and *mini-fiber* nodes in the network would be natural candidates for hosting the distribution tree node functionality.

In general, the root of the distribution tree may be different from the source content server. For example, a proxy or gateway server in a corporate intranet, or a cable headend may receive streaming video from the remote content server, and simultaneously stream out the video to its child nodes on the tree. The distribution tree itself may span multiple network service provider domains. The video service provider might build its distribution tree across multiple domains through cooperative contracts and peering agreements with the various network service providers.

Different parts of the internetwork may have different bandwidth capacities and/or may be carrying different traffic loads, or presenting different service guarantees, effectively providing different capacities for carrying multicast video traffic. The internal nodes in the internetwork may also have different amounts of resources, e.g. different buffer sizes for temporal caching of video frames. Similarly, clients may have different connectivity to the network, e.g., a client could be connected via a slow modem or high speed LAN. A client can be a workstation, PC, hand-held multimedia device or a set-top box connected to a television set, thus possessing varying buffering and computational resources.

The key question that we ask in this paper is how to design effective multicast smoothing solutions that can efficiently transmit video data over an internetwork to a set of clients, where the internetwork or the clients can be constrained either by buffer size availability or bandwidth capacity availability. We use results from studies in delivering streaming video to a single client [4, 5, 11, 12] that demonstrate the effectiveness of *workahead* smoothing. For prerecorded video, such workahead smoothing typically involves computing upper and lower constraints on the amount of data that can be transmitted at any instant, based on *a priori* knowledge of frame sizes and the size of the client playback buffer. A bandwidth smoothing algorithm is then used to construct a transmission schedule that minimizes burstiness subject to these constraints.

Our approach generalizes the server to single client bandwidth smoothing solution [11], where smoothing is done over a tandem set of nodes, to multicasting of smoothed streaming video over a distribution tree. We propose the idea of constructing a globally optimal set of smoothed transmission schedules, one per link of the distribution tree, depending upon the resource constraint (either buffer or bandwidth). The optimality metric is with respect to several criteria including minimization of the peak rate, the rate variability, and the effective bandwidth of the video.

2.1 Differential Caching

We propose using buffering at the root and intermediate nodes of the distribution tree to smooth the streaming video. Buffer availability at a node allows temporal caching of portions of the video streams. We refer to this technique as *differential caching* which can be of tremendous benefit to streaming video transmission in a

heterogeneous internetwork. Caching at the root node allows it to smooth an incoming live (or stored) video stream, and transmit the resultant schedule to a downstream node. The buffers at the internal nodes are used for several functions. First, these buffers allow a node to accommodate the differences between its incoming and outgoing transmission schedules when the outgoing and incoming link capacities are different. Second, as described in later sections, by increasing the *effective* or *virtual* smoothing buffer size for the upstream node, these buffers provide more aggressive smoothing opportunity along the upstream link. Finally, when the children of a node have different (effective) buffer capacities, the smoothed transmission schedules are different along the different outgoing links. For a given incoming transmission schedule, the node buffer must be used to temporally cache video frames till they are transmitted along each outgoing link. Thus differential caching allows the transmission schedule to each child to be decoupled to varying degrees depending on the size of the parent's buffer cache. This can be extremely beneficial from a network resource requirement point of view. For example, the parent node can smooth more aggressively to a child which has a larger (effective) buffer relative to a child with a smaller buffer. Differential caching allows the more constrained child to be served at its suitable pace, without requiring the frames to be retransmitted from higher up in the tree.

2.2 Application-aware Multicast

Our multicasting approach utilizes application-level information such as video frame sizes, and system resource availability, and delivers real-time streaming video data to clients for continuous playback. The notion of multicasting presented in this paper is somewhat distinct from traditional network-level multicast (e.g., IP multicasting [13]). IP multicasting techniques are not concerned with the real-time nature of the data or with maintaining streaming playback for all the clients. In addition, in traditional network-level multicast (e.g., IP multicast), each node forwards one copy of every relevant IP packet on each downstream path. In our approach, in addition to such packet duplication and forwarding, a node in the distribution tree also performs transmission schedule computation, differential caching, and real time streaming of video according to smoothed transmission schedules. Some researchers [14] have proposed using IP multicasting to transmit video data to multiple clients. IP multicast currently lacks native support for handling system heterogeneity (either in the clients or the network). In addition, in an environment such as today's Internet, large parts of the network are yet to become native multicast capable. Our application-aware approach can be implemented on top of network level multicast primitives, where they exist, and use unicast communication between nodes in the distribution tree elsewhere. Finally, our approach, by integrating smoothing with multicasting shows far superior performance with respect to reducing the network bandwidth requirements, compared to multicasting the unsmoothed video, as shown in Section 7.

Given a heterogeneous distribution tree environment, the important questions we need to address include (a) how to allocate resources to the intermediate nodes in the distribution tree so that streaming transmission is possible to all the clients ? (b) Given a particular resource allocation to the distribution tree, what transmission schedules should be used for the multicast, so that the bandwidth requirements are minimized? The rest of

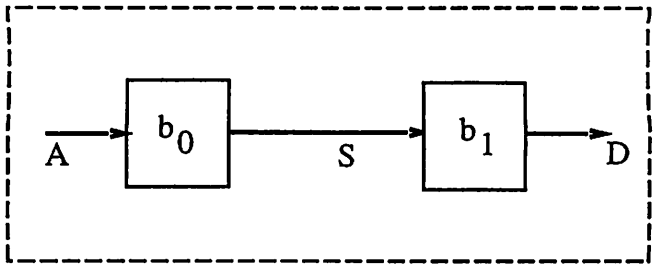


Figure 2: **Single Link Smoothing Model:** The smoothing server has a b_0-bit buffer and transmits the video to a smoothing client with a b_1-bit buffer. The video arrives according to an arrival vector **A**, is scheduled according to a vector **S**, and is played at the client according to a vector **D**, with a w-frame smoothing delay.

the paper is devoted to these questions. We start with an overview of unicast single link smoothing to provide necessary background for the rest of the paper.

3 Overview of Single Link Smoothing

This section describes the single link smoothing model, and overviews some important concepts and results which are crucial to deriving solutions for the multicast scenario.

A multimedia server can substantially smooth the bursty bandwidth requirements of streaming video by transmitting frames into the client playback buffer in *advance of* each burst. For prerecorded video, such workahead smoothing typically involves computing upper and lower constraints on the amount of data that can be transmitted at any instant, based on *a priori* knowledge of frame sizes and the size of the client playback buffer. A bandwidth smoothing algorithm is then used to construct a transmission schedule that minimizes burstiness subject to these constraints.

3.1 Single Link Model

Consider (Figure 2) an N-frame video stream, where the size of frame i is f_i bits, $i = 1, 2, \ldots, N$. This stream is transmitted across the network via a *smoothing server* node [15] (which has a b_0 bit buffer) to a *smoothing client* node which has a b_1 bit playback buffer. Without loss of generality, we assume a discrete time-model where one time unit corresponds to the time between successive frames. For a 30 frames/second full motion video, one time unit corresponds to $1/30^{th}$ of a second. *In the rest of the paper, any time index is assumed to be an integer.*

As shown in Figure 2, the video stream arrives at the server buffer b_0 according to an *arrival vector* $\mathbf{A} = (A_0, A_1, \ldots, A_N)$, where A_i is the cumulative amount of data which has arrived at the smoothing server by time $i, 0 \leq i \leq N + w$. The video stream is played back at the client according to a *(client) playback vector* $\mathbf{D} = \{D_0, D_1, \ldots, D_N\}$ where $D_0 = 0$ and for $i = 1, \ldots, N$, $D_i = \sum_{j=1}^{i} f_j$ is the cumulative amount of data

retrieved from the client buffer b_1 by i time unit since the start of playback. The video stream is transmitted from the server to the client according to a *transmission schedule* $\mathbf{S} = (S_0, S_1, \ldots, S_N)$, where S_i, $0 \leq i \leq N$, denotes the cumulative amount of data transmitted by the server by the i time unit. We will refer to the case where the server has an "infinite" buffer, in the sense that $b_0 \geq D_N$, as the *infinite source* model, whereas the case where $b_0 < D_N$ is referred to as the *finite source* model.

In general, the time the server starts transmitting a video stream (or the time when the video stream begins arriving at the smoothing server) may be different from the time the client starts playing back the video stream. The difference between these two start points is referred to as the *startup delay*. For a given startup delay $w \geq 0$, if we take the time the server starts transmitting a video stream as the reference point (i.e., time 0), then the playback vector $\mathbf{D}$ at the client will be shifted w time units to the right. This shifted playback vector is represented by $\mathbf{D}(w) = (D_0(w), D_1(w), \ldots, D_{N+w}(w))$, where $D_i(w) = 0$ for $i = 0, \ldots, w$, and $D_i(w) = D_{i-w}$ for $i = w+1, \ldots, N+w$. In this case, a server has $N + w + 1$ time units to transmit the video stream, namely, $\mathbf{S} = (S_0, S_1, \ldots, S_{N+w})$. For ease of notation, we will also extend the arrival vector $\mathbf{A}$ (which starts at time 0) with w more elements, namely, $\mathbf{A} = (A_0, A_1, \ldots, A_N, \ldots, A_{N+w})$, where for $i = N+1, \ldots, N+w$, $A_i = A_N$. Clearly, we must have $D_i(w) \leq S_i \leq A_i$ for $i = 0, 1, \ldots, N+w$. If we take the time the client starts the playback as the reference point (i.e., time 0), then we shift the arrival vector $\mathbf{A}$ w time units to the left. The corresponding arrival vector is denoted by $\mathbf{A}(-w) = (A_{-w}(w), A_{-w+1}(w), \ldots, A_N(w))$ where for $i = -w, \ldots, N-w$, $\mathbf{A}_i(w) = A_{i+w}$, and for any $i = N-w+1, \ldots, N$, $A_i(w) = A_N$. Similarly, the server starts transmission at time $-w$ according to a schedule $\mathbf{S} = (S_{-w}, \ldots, S_0, \ldots, S_N)$. In the rest of the paper, *we will take the time the server starts transmission as time 0 unless otherwise stated.* Depending on the context, $\mathbf{A}$ (or $\mathbf{D}$) will denote either the generic arrival (or playback) vector or the appropriately shifted version.

3.2 Buffer Constrained Single Link Optimal Smoothing

Given the above problem setting, we next overview the *buffer constrained single link smoothing* problem. Here the server and client buffers are the limiting resources, and the smoothing problem involves computing transmission schedules which can transmit the video from the server to the client in such a way as to reduce the variability of the transmitted video stream, thereby making efficient use of the network bandwidth.

To ensure lossless, continuous playback at the client, the server must transmit sufficient data to avoid buffer *underflow* at the client without *overflowing* the server buffer. This imposes a lower constraint, L_t, on the cumulative amount of data that the server can transmit by any time t, $0 \leq t \leq N + w$. We have

$$L_t = \max\{D_t(w), A_t - b_0\}$$

On the other hand, in order to prevent *overflow* of the client playback buffer(of size b_1 bits), the cumulative amount of data received by the client by time t cannot exceed $D_{t-1}(w) + b_1$. Denoting the upper constraint on the cumulative amount of data that the server can transmit by any time t by U_t, we have

$$U_t = \min\{D_{t-1}(w) + b_1, A_t\}, \quad 0 \leq t \leq N + w$$

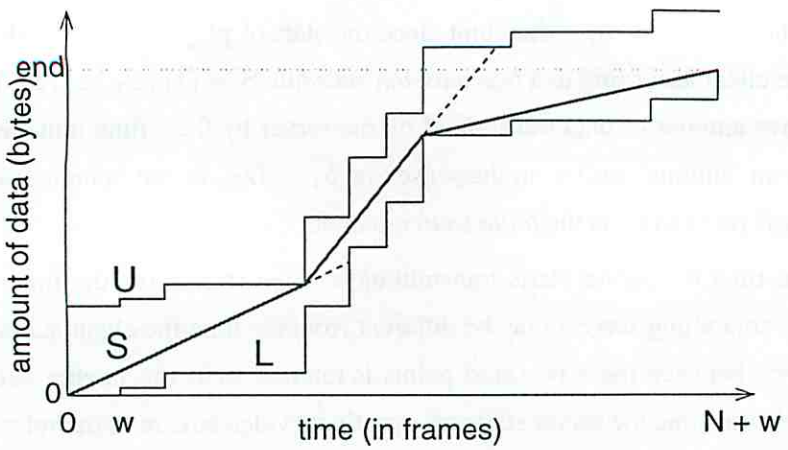


Figure 3: This figure shows an example of a transmission schedule $\mathbf{S}$ that stays between the upper and lower constraint vectors $\mathbf{U}$ and $\mathbf{L}$.

Given these lower and upper constraint vectors $\mathbf{L} = (L_0, \ldots, L_{N+w})$ and $\mathbf{U} = (U_0, \ldots, U_{N+w})$, a transmission schedule $\mathbf{S} = (S_0, S_1, \ldots, S_{N+w})$ is said to be *feasible* with respect to $\mathbf{L}$ and $\mathbf{U}$ if $S_0 = L_0$, $S_{N+w} = L_{N+w}$, and $\mathbf{L} \leq \mathbf{S} \leq \mathbf{U}$. In other words, a feasible schedule $\mathbf{S}$ always stays between $\mathbf{L}$ and $\mathbf{U}$ (see Figure 3). Thus it neither underflows, nor overflows the server or client buffer.

In general, for a given pair of constraint vectors $(\mathbf{L}, \mathbf{U})$ such that $\mathbf{L} \leq \mathbf{U}$, more than one feasible transmission schedules $\mathbf{S}$ exist. Let $\mathcal{S}(\{b_i\}, \mathbf{A}, \mathbf{D})$ denote the set of all feasible schedules with respect to $\mathbf{L}$ and $\mathbf{U}$. Among all feasible schedules, we would like to find a "smoothest" schedule that minimizes network utilization according to some performance metrics [5]. In [16], a measure of smoothness based on the theory of *majorization* is proposed, and the resulting *optimally* smoothed schedule minimizes a wide range of bandwidth metrics such as the peak rate, the variability of the transmission rates as well as the empirical effective bandwidth. Henceforth, we shall refer to the "optimal schedule" constructed in [16] as the *majorization schedule*.

For any two K-dimensional real vectors $X = (X_1, \ldots, X_K)$ and $Y = (Y_1, \ldots, Y_K)$, Y is said to *majorize* X (denoted by $X \prec Y$) [17] if $\sum_{i=1}^{K} X_i = \sum_{i=1}^{K} Y_i$, and for $k = 1, \ldots, K-1$, $\sum_{i=1}^{k} X_{[i]} \leq \sum_{i=1}^{k} Y_{[i]}$ where $X_{[i]}$ (resp., $Y_{[i]}$) is the i-th largest component of X (resp., Y). A notion closedly associated with majorization is *Schur-convex* function. A function $\phi : I\!R^K \rightarrow I\!R$ is said to be a *Schur-convex* function iff $X \prec Y \Rightarrow \phi(X) \leq \phi(Y), \forall X, Y \in I\!R^K$. Examples of Schur-convex functions include $\phi(X) = \max_i X_i$, and $\phi(X) = \sum_{i=1}^{K} f(X_i)$ where f is any convex real function.

In the context of video smoothing, for a transmission schedule $\mathbf{S} = (S_0, S_1, \ldots, S_{N+w})$, define $\mathbf{R}(\mathbf{S}) = (S_1 - S_0, \ldots, S_{N+w} - S_{N+w-1})$. A schedule $\mathbf{S}_1$ is majorized by (or intuitively, "smoother" than) another schedule $\mathbf{S}_2$ (denoted by $\mathbf{S}_1 \prec \mathbf{S}_2$) if $\mathbf{R}(\mathbf{S}_1) \prec \mathbf{R}(\mathbf{S}_2)$. The *majorization schedule*, $\mathbf{S}^*(\mathbf{L}, \mathbf{U})$, or in short $\mathbf{S}^*$, is the schedule such that $\mathbf{R}(\mathbf{S}^*) \prec \mathbf{R}(\mathbf{S}), \forall \mathbf{S} \in \mathcal{S}(\{b_i\}, \mathbf{A}, \mathbf{D})$. In particular, the peak rate of $\mathbf{S}^*$, $peak(\mathbf{S}^*) = \max_k \{S_k - S_{k-1}\}$, is minimal among all feasible schedules $\mathbf{S}$. In [16], it is shown that $\mathbf{S}^*$ exists and is unique (provided that $\mathbf{L} \leq \mathbf{U}$), and it can be constructed in linear time.

For a given majorization schedule $\mathbf{S}^* = (S_0^*, S_1^*, S_2^*, \ldots, S_{N+w}^*)$, we say k, $1 \leq k \leq N + w - 1$, is a *change point* of the schedule if $S_{k+1}^* - S_k^* \neq S_k^* - S_{k-1}^*$. Moreover, k is said to be a *convex change point* if $S_{k+1}^* - S_k^* \geq S_k^* - S_{k-1}^*$, and a *concave change point* if $S_{k+1}^* - S_k^* \leq S_k^* - S_{k-1}^*$. A key feature of the majorization schedule is summarized by the following lemma, which is a simple byproduct of the majorization proof in [16].

Lemma 3.1 *Let* $\mathbf{S}^*$ *be the majorization schedule for a given pair of the lower and upper constraint vectors* $\mathbf{L}$ *and* $\mathbf{U}$ *where* $\mathbf{L} \leq \mathbf{U}$. *If* k *is a concave change point of* $\mathbf{S}^*$, *then* $S_k^* = L_k$; *and if* k *is a convex change point of* S_k, *then* $S_k^* = U_k$.

Based on this lemma, the following *domination properties* of the majorization schedules with respect to different buffer size are established in [18]. These lemmas will be used extensively in the rest of paper to construct optimal smooth schedules for a distribution tree of smoothing servers.

Lemma 3.2 *Let* $\mathbf{L}_1 \leq \mathbf{U}_1$ *and* $\mathbf{L}_2 \leq \mathbf{U}_2$ *be such that* $\mathbf{L}_1 \leq \mathbf{L}_2$ *and* $\mathbf{U}_1 \leq \mathbf{U}_2$. *Then* $\mathbf{S}_1^* \leq \mathbf{S}_2^*$. *Moreover, if* $\mathbf{L}_1 = \mathbf{L}_2$, *then any concave (resp., convex) change point of* $\mathbf{S}_2$ *is also a concave (resp., convex) change point of* $\mathbf{S}_1$.

In particular, we have

Lemma 3.3 *For any* $b_2 \geq b_1 > 0$ *and* $\mathbf{L}_1 = \mathbf{L}_2$, *define* $\mathbf{U}_1 = \mathbf{L}_1 + vec(b_1)$ *and* $\mathbf{U}_2 = \mathbf{L}_2 + vec(b_2)$, *where* $vec(a)$ *denotes a vector whose components are all equal to* a. *Then the set of the change points of* $\mathbf{S}_1^*$ *is a superset of the change points of* $\mathbf{S}_2^*$.

3.3 Rate Constrained Single Link Optimal Smoothing

We next consider the dual problem to the buffer constrained single link optimal smoothing problem: the *rate constrained* single link optimal smoothing problem, where the bandwidth of the link connecting the server and the client is constrained by a given rate r. Here we describe results which we will use to develop optimal solutions for the rate-constrained multicast scenario. Due to the rate constraint, the buffer at the client must be *sufficient large* to ensure continuous video playback at the client. Furthermore, it may be necessary for the server to start video transmission *sufficiently early*. Hence *the rate constraint imposes both a minimum buffer requirement and startup delay at the client*. In this context, a server transmission schedule is *feasible* if the transmission rate of the schedule never exceeds the rate constraints at any time (as well as the arrival vector if any) and the amount of data needed for client playback is always satisfied at any time. We are again interested in finding the "smoothest" transmission schedule among all the feasible schedules for a given rate constraint. In order to solve this rate constrained single link optimal smoothing problem, we need to address the following two basic questions:

1. what is the minimum start-up delay between the server and the client so that a feasible transmission schedule exists?

2. among all feasible schedules, what is the smallest client buffer necessary for feasible transmission ?

We assume that the client playback starts at time 0, and as before, the cumulative client playback vector is denoted by $\mathbf{D}(0) = (D_0(0), D_1(0), \ldots, D_N(0))$. For any $w \geq 0$, if the server starts transmission at time $-w$, the transmission schedule is represented as a vector $\mathbf{S} = (S_{-w}, S_{-w+1}, \ldots, S_N)$, where as before, S_k denotes the (cumulative) amount of data transmitted by the server by time k. (For reasons to be clear later, we also assume that $S_{k'} = 0$ for any $k \leq w$.) Given a startup delay w, we assume that the arrival vector starts at time $-w$ instead of time 0, with the shifted arrival vector $\mathbf{A}(-w) = (A_{-w}(w), A_{-w+1}(w), \ldots, A_N(w))$. Consider the *infinite source* model. It is possible to construct a *feasible* transmission schedule $\mathbf{S}^{late}$ [19–21], which transmits data as late as possible, while obeying rate constraint r. We refer to this as the *lazy schedule* (Figure 4).

The *lazy transmission schedule* $\mathbf{S}^{late}$ is defined recursively from time N backwards as follows:

$$S_k^{late} = \begin{cases} D_N(0), & k = N, \\ \max\{S_{k+1}^{late} - r, D_k(0)\}, & \text{for } 0 \leq k < N, \\ \max\{S_{k+1}^{late} - r, 0\}, & \text{for } k < 0, \end{cases} \tag{1}$$

From this definition, it is clear that the client is never starved (i.e., $S_k \geq D_k(0)$, $k = 0, 1, \ldots, N$), and that $peak(\mathbf{S}^{late}) \leq r$. In the case that $peak(\mathbf{S}^{late}) < r$, then $\mathbf{S}^{late} = \mathbf{D}(0)$. Namely, $\mathbf{S}^{late}$ is exactly the same as the client playback vector $\mathbf{D}(0)$. In general, $\mathbf{S}^{late}$ is composed of two types of segments: segments $[t_1, t_2]$ where $\mathbf{S}^{late}$ follows the client playout vector $\mathbf{D}(0)$, i.e., $S_k^{late} = D_k(0)$, for $k \in [t_1, t_2]$; and segments $[t_1, t_2]$ where transmission rate is exactly r, i.e., $S_{t_1}^{late} = D_{t_1}(0)$, $S_{t_2}^{late} = D_{t_2}(0)$, and for any $k \in (t_1, t_2)$, $S_k^{late} = D_{t_1}(0) + r * (k - t_1)$. For convenience, we refer to a segment of $\mathbf{S}^{late}$ whose transmission rate equals $peak(\mathbf{S}^{late})$ as a *peak rate segment*. If $peak(\mathbf{S}^{late}) = r$, then any segment of the second type mentioned above is a peak rate segment. Finally, we note that the definition (1) directly leads to a linear time algorithm for constructing $\mathbf{S}^{late}$.

Now define

$$b^*(r) = \max_{k \geq 0}\{S_k^{late} - D_k(0)\} \tag{2}$$

and

$$w^*(r, \mathbf{A}) = \min\{w \geq 0 : A_k(-w) - S_k^{late} \geq 0, -w \leq k \leq N\} \tag{3}$$

Clearly, $b^*(r)$ is the mimimum buffer required at the client for the transmission schedule $\mathbf{S}^{late}$ without incurring loss of data, and $w^*(r, \mathbf{A})$ is the minimum start-up delay with respect to which $\mathbf{S}^{late}$ conforms to the arrival vector, namely, $S_k \leq A_k(-w^*)$, for $k \geq w^*$. As a result, $\mathbf{S}^{late}$ is a *feasible* schedule. We will write b^* for

$b^*(r)$ and w^* for $w^*(r, \mathbf{A})$ whenever there is no danger of confusion. The following results are important for developing solutions for the multicast scenario.

It can be shown (for a proof see [22]) that b^* is the minimal buffer requirement and w^* is the minimal start-up delay among all feasible schedules with respect to the given rate constraint r and arrival vector $\mathbf{A}$.

Theorem 3.1 *Let $\mathcal{S}(r, \mathbf{A}, \mathbf{D})$ be the set of all feasible schedules with respect to the rate constraint r and the arrival vector $\mathbf{A}$. For any $\mathbf{S} \in \mathcal{S}(r, \mathbf{A}, \mathbf{D})$, let $b(\mathbf{S})$ and $w(\mathbf{S})$ denote the minimum buffer requirement and minimum start-up delay of $\mathbf{S}$. We have*

$$b^*(r) = \min_{\mathbf{S} \in \mathcal{S}(r,\mathbf{A},\mathbf{D})} b(\mathbf{S}) \quad \textit{and} \quad w^*(r, \mathbf{A}) = \min_{\mathbf{S} \in \mathcal{S}(r,\mathbf{A},\mathbf{D})} w(\mathbf{S}). \tag{4}$$

Analogous to Lemma 3.2, the following property holds for the lazy schedule $\mathbf{S}^{late}$.

Lemma 3.4 *Given two rate constraints r_i, $i = 1, 2$ such that $r_1 > r_2$. For $i = 1, 2$, let $\mathbf{S}_i^{late}$ be the lazy schedule with respect to the rate constraint r_i. Then*

$$\mathbf{S}_1^{late} \leq \mathbf{S}_2^{late}. \tag{5}$$

Proof: Suppose (5) does not hold. Then there exists k, $k \leq N$, $S_{2,k}^{late} < S_{1,k}^{late}$. From the definition of the lazy schedule, k must belong to a peak rate segment of $\mathbf{S}_1^{late}$ (as otherwise $S_{1,k}^{late} = D_k(0) \leq S_{2,k}^{late}$, contradicting our supposition). Let $l > k$ be the earliest time after k such that $S_{1,l}^{late} = D_l(0)$, i.e., l is the right endpoint of the peak rate segment (note that l always exists, as $S_N^{late} = D_N(0)$). Then $S_{1,l}^{late} = S_{1,k}^{late} + r_1 * (l - k)$. On the other hand, we have $S_{2,l}^{late} \geq D_l(0)$. Hence, the minimum rate of $\mathbf{S}_2^{late}$ during the time interval (k, l) is $(S_{2,l}^{late} - S_{2,k}^{late})/(l - k) \geq (S_{1,l}^{late} - S_{1,k}^{late})/(l - k) = r_1 > r_2$. This contradicts the fact that $\mathbf{S}_2^{late}$ obeys the rate constraint r_2. Therefore we must have $S_{2,k}^{late} \geq S_{1,k}^{late}$, for all $k \leq N$. ∎

The following important property regarding $b^*(r)$ and $w^*(r, \mathbf{A})$ follows directly from Lemma 3.4 and the definitions of $b^*(r)$ and $w^*(r, \mathbf{A})$.

Corollary 3.1 *The minimum buffer size $b^*(r)$ and minimum startup delay $w^*(r, \mathbf{A})$ are nonincreasing functions of the rate constraint r.*

We now proceed with the problem of finding the "optimally smoothed" schedule under the rate-constrained setting. For ease of exposition, choose time 0 as the time A_0 arrives at the server and the server starts video transmission. Corresponding to any playback startup delay, w, where $w \geq w^*$, the client playback vector is $\mathbf{D}(w)$, i.e., $\mathbf{D}(0)$ shifted w time units to the right. The new lazy schedule $\mathbf{S}^{late}$ is then defined with respect to $\mathbf{D}(w)$, and is the original $\mathbf{S}^{late}$ shifted w time units to the right. By the definition of w^*, we have $w^* = \min\{w > 0 : A_k - S_k^{late} \geq 0, 0 \leq k \leq N + w\}$. The following theorem relates the rate-constrained optimal smoothing problem to its dual buffer-constrained optimal smoothing problem.

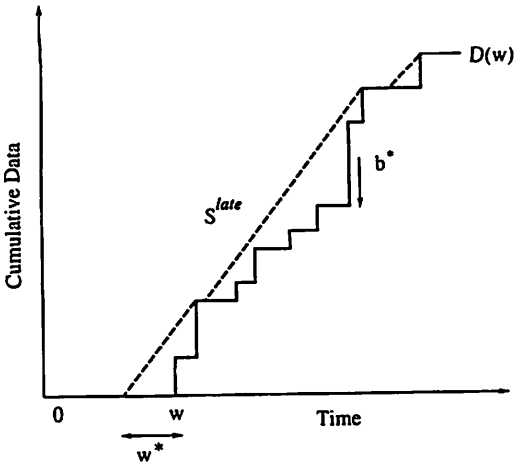


Figure 4: **Rate Constrained Smoothing**: This figure shows the transmission schedule $\mathbf{S}^{late}$ which transmits data as late as possible, while obeying rate constraint r. b^* and w^* are the respective minimum buffer and minimum startup delays necessary to satisfy rate constraint r.

Theorem 3.2 *For a given rate constraint r and an arrival vector $\mathbf{A}$, let $b^*(r)$ and $w^*(r, \mathbf{A})$ be the minimum client buffer requirement and startup delay of $S(r, \mathbf{A}, \mathbf{D})$, as defined in Theorem 3.1. For any $w \geq w^*(r, \mathbf{A})$, define*

$$\mathbf{L} = \mathbf{D}(w) \text{ and } \mathbf{U} = \min\{\mathbf{A}, \mathbf{D}(w) + vec(b^*(r))\}. \tag{6}$$

Then the majorization schedule $\mathbf{S}^*$ with respect to the buffer-constraints $\mathbf{L}, \mathbf{U}$ is also a feasible schedule with respect to the rate-constraint r, and $peak(\mathbf{S}^*) = peak(\mathbf{S}^{late})$. In particular, if $peak(\mathbf{S}^{late}) = r$, then $peak(\mathbf{S}^*) = r$.

Proof: From the definition of $\mathbf{S}^{late}$, $b^*(r)$ and $w^*(r, \mathbf{A})$, it is clear that $\mathbf{L} \leq \mathbf{S}^{late} \leq \mathbf{U}$, i.e., $\mathbf{S}^{late} \in S(b^*, \mathbf{A}, \mathbf{D})$. Therefore, $\mathbf{S}^* \prec \mathbf{S}^{late}$. As a result, $peak(\mathbf{S}^*) \leq peak(\mathbf{S}^{late})$. Hence, $\mathbf{S}^* \in S(r, \mathbf{A}, \mathbf{D})$, i.e., $\mathbf{S}^*$ is a feasible schedule with respect to the rate constraint r.

We now show that $peak(\mathbf{S}^*) = peak(\mathbf{S}^{late})$. Let x be such that the difference between $\mathbf{S}^{late}$ and the client playback vector $\mathbf{D}(w)$ is maximized at time x, i.e., $S_x^{late} - D_x(w) = b^*$. We first argue that $x \geq w$. Note that for $t \in [0, w)$, $D_t(w) = 0$, within this interval the buffer occupancy is an increasing function of t (see Figure 4). Hence $x \geq w$. The segment of $\mathbf{S}^{late}$ containing x must therefore be a peak rate segment. Let $y > x$ be the right endpoint of this peak rate segment. We have $S_x^{late} = D_x(w) + b^*$ and $S_y^{late} = D_y(w) = S_x^{late} + (y - x) * peak(\mathbf{S}^{late})$. As $S_x^* \leq S_x^{late}$ and $S_y^* \geq S_y^{late}$, we conclude that the transmission rate of $\mathbf{S}^*$ must be at least equal to $peak(\mathbf{S}^{late})$ during some part of the segment $[x, y]$. This can only happen when $\mathbf{S}^*$ coincides with $\mathbf{S}^{late}$ within this segment. We therefore establish that $peak(\mathbf{S}^*) = peak(\mathbf{S}^{late})$.

■

As a consequence of Theorem 3.2, we see that for $w = w^*$, the majorization schedule $\mathbf{S}^*$ is majorized (thus "smoothest" under the measure of majorization) by any feasible schedule in $\mathcal{S}(r, \mathbf{A}, \mathbf{D})$ which has the *same* client buffer requirement $b^*(r)$ and $w^*(r, \mathbf{A})$. In the following, we establish an important property for the majorization schedule in the context of the rate constrained smoothing problem. This will be useful for computing the optimal buffer allocation in the rate-constrained multicast scenario.

Lemma 3.5 *Given two rate constraints r_i, $i = 1, 2$, where $r_1 \geq r_2$, let $b_i^* = b^*(r_i)$ and $w_i^* = w^*(r_i, \mathbf{A})$ be the corresponding minimum client buffer requirement and startup delay with respect to the rate constraint r_i and an arrival vector $\mathbf{A}$). For any $w \geq \max\{w_1^*, w_2^*\}$, define $\mathbf{L}_i = \mathbf{D}(w)$ and $\mathbf{U}_i = \min\{\mathbf{A}, \mathbf{D}(w) + vec(b_i^*)\}$, $i = 1, 2$. Let $\mathbf{S}_i^*$ be the majorization schedule with respect to $(\mathbf{L}_i, \mathbf{U}_i)$, $i = 1, 2$. Then*

$$\mathbf{S}_1^* \leq \mathbf{S}_2^* \text{ and } \max\{\mathbf{S}_2^* - \mathbf{S}_1^*\} = b_2^* - b_1^*. \tag{7}$$

Moreover, for any feasible schedule $\mathbf{S}_i$ such that $\mathbf{L}_i \leq \mathbf{S}_i \leq \mathbf{U}_i$ and $peak(\mathbf{S}_i) \leq peak(\mathbf{S}_i^*)$, we have

$$\max\{\mathbf{S}_2 - \mathbf{S}_1\} \geq b_2^* - b_1^* \tag{8}$$

Proof: As $b_1^* \leq b_2^*$, we have

$$\mathbf{U}_1^* \leq \mathbf{U}_2^* \leq \mathbf{U}_1^* + vec(b_2^* - b_1^*) \text{ and } \mathbf{L}_1^* \leq \mathbf{L}_2^* \leq \mathbf{L}_1^* + vec(b_2^* - b_1^*)$$

Applying Lemma 3.2 yields

$$\mathbf{S}_1^* \leq \mathbf{S}_2^* \leq \mathbf{S}_1^* + vec(b_2^* - b_1^*) \tag{9}$$

The second inequality above implies that $\max\{\mathbf{S}_2^* - \mathbf{S}_1^*\} \leq b_2^* - b_1^*$. We now show that the equality is attained at some time. From the proof of Theorem 3.2, we see that there exists t_1 and t_2, $t_1 < t_2$ such that $S_{2,t_1}^* = D_{t_1}(w) + b_2^*$, $S_{2,t_2}^* = D_{t_2}(w)$, and for any $t \in (t_1, t_2)$, $S_{2,t}^* = S_{2,t_1}^* + peak(\mathbf{S}_2^*) * (t - t_1)$. On the other hand, $S_{1,t_1}^* \geq D_{t_1}(w)$. The second inequality in (9), coupled with the fact that $S_{1,t_1}^* \leq D_{t_1}(w) + b_1^*$ yields that that $S_{1,t_1}^* = D_{t_1}(w) + b_1^*$. Hence, $S_{2,t_1}^* - S_{1,t_1}^* = b_2^* - b_1^*$. This completes the proof of (7).

In order to prove (8), we follow the same line of argument. As in the proof of Theorem 3.2, we observe that any feasible schedule $\mathbf{S}_2$ such as $peak(\mathbf{S}_2) \leq peak(\mathbf{S}_2^*)$ must follow the same transmission schedule as $\mathbf{S}_2^*$ during $[t_1, t_2]$ (or the peak rate segment $[t_1, t_2]$ of $\mathbf{S}_2^{late}$). In other words, we have

$$S_{2,t_1} = D_{t_1}(w) + b_2^*, \quad S_{2,t_2} = D_{t_2}(w), \text{ and} \forall t \in (t_1, t_2), \quad S_{2,t} = S_{2,t_1} + peak(\mathbf{S}_2) * (t - t_1).$$

On the other hand, $S_{1,t_1} \leq D_{t_1}(w) + b_1^*$. Thus $S_{2,t_1} - S_{1,t_1} \geq b_2^* - b_1^*$. This completes the proof of the lemma. ∎

4 Multicast Distribution of Smoothed Video

In this section we first present a formal model for the video multicast distribution tree. We then outline the buffer and rate constrained multicast smoothing problems.

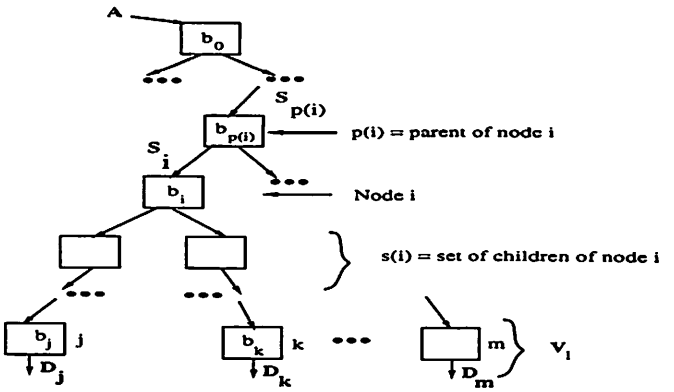


Figure 5: **Multicast Smoothing Model:** The video arrives according to an arrival vector $\mathbf{A}$, is scheduled according to a vector $\mathbf{S}_i$ from node i's parent to node i, and is played at leaf i according to a vector $\mathbf{D}_i$. See text for explanation of notations.

4.1 Video Multicast Distribution Tree Model

Consider a directed tree $T = (V, E)$ (Figure 5) where $V = \{0, 1, \ldots, n\}$ is the set of nodes and E is the set of directed edges within the tree. Consider a node $i \in V$. Let $p(i)$ denote its parent and $s(i)$ the set of children attached to i, i.e., $(p(i), i) \in E$ and $s(i) = \{j \in V : (i, j) \in E\}$. We will take node 0 to be the root of the tree; it has no parent. Let $V_l \subset V$ be the set of leaves in the tree; it is defined as $V_l = \{i \in V : s(i) = \emptyset\}$. Note that node 0 could itself be the source server of the video, or, the video may be streaming into the root from a remote source. Also, a leaf can be an end-client, or an egress node in the distribution tree. We will use the two pairs of terms (*root* and *source*) and (*leaf* and *client*) interchangeably.

Associated with node $i \in V$ is a buffer of capacity $0 \leq b_i < \infty$. Consider a video stream which arrives at node 0 destined for all nodes $i \in V_l$. Associated with this video stream is an *arrival vector* $\mathbf{A} = (A_0, \ldots, A_N)$ in which the k-th component corresponds to the cumulative amount of data from the stream which has arrived at node 0 by time $k = 0, 1, \ldots, N$. It is assumed that $A_k \geq A_{k-1}$, $k = 1, \ldots, N$. For each node $i \in V_l$, $\mathbf{D}_i = (D_{i,0}, \ldots, D_{i,N})$ denotes the *client playout vector*[1] where $D_{i,k}$ represents the cumulative amount of data from this stream that must be removed at leaf i by time $k = 0, 1, 2, \ldots, N$. It is assumed that $D_{i,0} = 0$ and $D_{i,k} \geq D_{i,k-1}$, $k = 1, \ldots, N$. Associated with each node $i \neq 0$, $i \in V$ is a *schedule* $\mathbf{S}_i = (S_{i,0}, \ldots, S_{i,N})$ in which the k-th component denotes the cumulative amount of data transmitted by node $p(i)$ to node i by time $k = 0, 1, \ldots$, $i = 1, \ldots, n$. Note that $\mathbf{S}_0 \equiv \mathbf{A}$, and $S_{i,k} \geq S_{i,k-1}$, $1 \leq k \leq N$. A set of schedules $\{\mathbf{S}_i\}_{i \in V}$ is said to be *feasible* if, simultaneously using the component schedules in the set for video distribution along the tree does not violate any system constraints, and results in lossless, starvation-free playback at each leaf.

We will find it useful to let $\mathbf{R}_i$ denote the vector $\mathbf{R}_i = (R_{i,1}, \ldots, R_{i,N})$ where $R_{i,k} = S_{i,k} - S_{i,k-1}$,

[1]Note that here we are *not* assuming that all clients have the same playback vectors. Namely, for $i, j \in V_l$, $i \neq j$, it is possible that $\mathbf{D}_i \neq \mathbf{D}_j$. This may be the case when clients playback the same video at different frame rates or when layered video encoding is employed.

$k = 1, \ldots, N$. The k-th component of $\mathbf{R}_i$ corresponds to the amount of data that must be transferred between nodes $p(i)$ and i during the time interval $[(k-1), k)$, $k = 1, \ldots, N$, $i = 1, \ldots, n$.

4.2 Buffer Constrained Optimal Multicast Smoothing

Given a set of buffer allocations $\{b_i\}_{i \in V}$, and consumption vectors $\{\mathbf{D}_i\}_{i \in V_l}$ at the leaves of the distribution tree, let $\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\})$ denote the set of all feasible schedules for this system. In this context, the set $\{\mathbf{S}_i\}_{i \in V}$ is feasible if

$$\max\{\mathbf{S}_{p(i)} - vec(b_{p(i)}), \max_{j \in s(i)} \mathbf{S}_j\} \leq \mathbf{S}_i \leq \min\{\min_{j \in s(i)} \mathbf{S}_j + vec(b_i), \mathbf{S}_{p(i)}\}, \quad i \in V \setminus V_l, \tag{10}$$

and

$$\max\{\mathbf{S}_{p(i)} - vec(b_{p(i)}), \mathbf{D}_i\} \leq \mathbf{S}_i \leq \min\{\mathbf{D}_i + vec(b_i), \mathbf{S}_{p(i)}\}, \quad i \in V_l, \tag{11}$$

where $\mathbf{S}_0 \equiv \mathbf{A}$ [2].

Intuitively, at node i, at any time k, the cumulative incoming data $S_{i,k}$ should be sufficient to satisfy the cumulative outgoing data to any child node, and must not exceed the cumulative data being received at its parent $p(i)$. Also, $\mathbf{S}_i$ should transmit data fast enough to prevent buffer overflow at $p(i)$, but not fill up the buffer at i so quickly that the transmission to some child $S_j (j \in s(i))$ is unable to transmit data from the buffer in time, before it gets overwritten.

The following inequalities follow almost immediately,

$$\mathbf{S}_i \leq \mathbf{S}_{p(i)} \leq \mathbf{S}_i + vec(b_i), \quad i \in V \setminus \{0\}. \tag{12}$$

Two important questions in this setting are

1. *Buffer feasibility* : Is the buffer allocation feasible, i.e., is the set of feasible schedules $\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\})$ nonempty? Note that the feasibility question arises as all the buffer allocations are assumed to be finite. Given a particular arrival vector $\mathbf{A}$ and playback vectors $\{\mathbf{D}_i\}_{i \in V_l}$ at the leaves, it is possible that no feasible transmission schedule is possible at one or more edges in the tree, due to buffer overflow at the source or sink of that edge.

2. *Optimal smoothed schedules* : For a feasible buffer allocation, what is the optimal set of smoothed transmission schedules $\{\mathbf{S}_i\}_{i \in V}$?

Finally, note that for the case $n = 1$, this system is reduced to the buffer constrained single link smoothing problem of Section 3 with $\mathbf{L} = \max(\mathbf{D}, \mathbf{A} - vec(b_0))$ and $\mathbf{U} = \min(\mathbf{D} + vec(b_1), \mathbf{A})$ and the $O(N)$ algorithm given in [16] determines the majorization schedule, $\mathbf{S}^*$, such that $\mathbf{S}^* \prec \mathbf{S}, \forall \mathbf{S} \in \mathcal{S}(\{b_i\}, \mathbf{A}, \mathbf{D})$.

[2] The expression $V \setminus V_l$ refers to all elements of set V except those belonging to set V_l

4.3 Rate Constrained Optimal Multicast Smoothing

We next consider the problem of rate constrained multicast smoothing. As in the case of the single link smoothing problem, we assume that the bandwidth of the links in a multicast video distribution tree is limited instead of buffering capacity at the nodes being constrained. Consider the *infinite source* model, i.e., $b_0 \geq D_N$.

Following the notation introduced earlier in the section, the client playback vector at node $i \in V_l$ is $\mathbf{D} = \{D_0, \ldots, D_N\}$, where D_i is the cumulative amount of data consumed by a client i time units after it starts playback. For $i \in V \setminus \{0\}$, a rate constraint r_i is associated with the link $(p(i), i)$, which specifies the bandwidth available on the link. $\mathbf{S}_i = (S_{i,0}, S_{i,1}, \ldots)$ denotes a schedule used by node $p(i)$ to transmit data on the link $(p(i), i)$ to node i, where $S_{i,k}$ is the cumulative amount of data transmitted by time k. We say $\{\mathbf{S}_i\}_{i \in V}$ is a *feasible* schedule if the following conditions are satisfied:

$$peak(\mathbf{S}_i) \leq r_i, \quad i \in V \setminus \{0\}, \tag{13}$$

$$\text{and} \qquad \max_{j \in s(i)} \mathbf{S}_j \leq \mathbf{S}_i \leq \mathbf{S}_{p(i)} \text{ if } i \in V - V_l, \text{ or } \mathbf{D}_i \leq \mathbf{S}_i \leq \mathbf{S}_{p(i)} \text{ if } i \in V_l. \tag{14}$$

We define $\mathcal{S}(T, \{r_i\}, \mathbf{A}, \mathbf{D})$ to be the set of feasible transmission schedule sets for this system. We are interested in addressing the following questions.

1. *Minimum startup delay* : What is the minimum (common) playback startup delay w^* for all the clients for which a feasible transmission schedule exists for the system, i.e., $\mathcal{S}(T, \{r_i\}, \mathbf{A}, \mathbf{D}) \neq \emptyset$? Due to the transmission rate constraints, some minimum startup delay may be required to build up sufficient data to guarantee starvation-free playback at each client. Here we assume that *all the clients start playback at the same time.*

2. *Optimal buffer allocation* : What is the minimum buffer allocation b_i at each node $i \in V \setminus \{0\}$ of the distribution tree, and what is the minimum total buffer allocation $b = \sum_{i \in V \setminus \{0\}} b_i$ among all feasible schedules for the system? As explained before, the buffer is used for differential caching. Observe that for a given set of feasible schedules $\{\mathbf{S}_i\}_{i \in V}$, the buffer allocation $\{b_i\}_{i \in V \setminus \{0\}}$ for the distribution tree is said to be *feasible* if the following constraints are satisfied: b_i must be sufficient large to ensure lossless video transmission, namely, it must be able to accommodate both (a) the maximum difference between the amount of data transmitted from node i to any child node l according a feasible schedule $\mathbf{S}_l$ and that from node i to another child node k according to a schedule $\mathbf{S}_k$, as well as (b) the maximum difference between the amount of data arriving at node i according to a schedule $\mathbf{S}_i$ and that transmitted from node i to a child node k according to a schedule $\mathbf{S}_k$. Formally,

$$b_i \geq \max_{l,k \in s(i)} \{\max\{\mathbf{S}_l - \mathbf{S}_k\}\}, \tag{15}$$

$$b_i \geq \max_{k \in s(i)} \{\max\{\mathbf{S}_i - \mathbf{S}_k\}\}. \tag{16}$$

5 Buffer Constrained Multicast Optimal Smoothing

In this section we develop solutions to the buffer constrained multicast problems outlined in Section 4.2. We utilize results for the buffer constrained single link smoothing problem, and exploit properties of majorization schedules to build our solutions for the multicast situation. Analogous to the single link case, a key step in our approach involves computing upper and lower constraint curves at individual nodes in the distribution tree. Unlike the single link case, the constraints at a node can be affected by both the constraints at its directly connected nodes *as well as* at remote nodes. We shall present some important results based on which we develop algorithmic solutions to the two questions posed in Section 4.2.

Upper Constraint:

For $i \in V$, we define a vector $\mathbf{U}_i^b$ recursively as follows:

$$\mathbf{U}_i^b = \begin{cases} \mathbf{D}_i + vec(b_i) \text{ for } i \in V_l \\ \min_{j \in s(i)} \mathbf{U}_j^b + vec(b_i) \text{ for } i \in V \setminus V_l \end{cases} \tag{17}$$

$\mathbf{U}_i^b$ can be thought of as the *buffer overflow* (or *upper*) *constraint vector* for the subtree rooted at node i when it is fed by the source of the prerecorded video, i.e., $A_k = A_N, k = 0, \ldots, N$. We will see shortly that it is possible that a buffer constraint somewhere upstream in the tree may impose a more stringent constraint than $\mathbf{U}_i^b$ on the transmission schedules for the subtree rooted at node i.

For $i \in V$, let $P(i)$ denote the set of nodes on the path from the root, 0, to node i and define

$$\mathbf{U}_i^c = \min_{j \in P(i)} \mathbf{U}_j^b \tag{18}$$

We will refer to $\mathbf{U}_i^c$ as the *effective buffer overflow constraint vector* of the subtree rooted at i. Observe that $\mathbf{U}_i^c \leq \mathbf{U}_i^b$. The effective overflow vectors exhibit the following properties.

$$\mathbf{U}_i^c \leq \mathbf{U}_{p(i)}^c \leq \mathbf{U}_i^c + vec(b_{p(i)}). \tag{19}$$

The first of these inequalities comes from the definition of $\mathbf{U}_i^c$ and the observation that $P(p(i)) \subset P(i)$. The second inequality takes a little more work involving the definition of $\mathbf{U}_j^b$.

Lower Constraint:

We can associate a playout vector $\mathbf{D}_i$ with an interior node $i \in V \setminus V_l$, namely $\mathbf{D}_i = \max_{j \in s(i)} \mathbf{D}_j$. Also associated with node i is an *effective buffer underflow constraint vector*, $\mathbf{L}_i^c$ defined by the following recurrence relation:

$$\mathbf{L}_i^c = \begin{cases} \mathbf{D}_i, & i = 0 \\ \max(\mathbf{D}_i, \mathbf{L}_{p(i)}^c - vec(b_{p(i)})), & i \neq 0. \end{cases} \tag{20}$$

We now consider a single link system with an arrival vector $\mathbf{A}$ in which the source has a buffer capacity of size $G_{p(i)} \equiv \sum_{j \in P(p(i))} b_j$ and the receiver has buffer overflow and underflow constraints $\mathbf{U}_i^c$ and $\mathbf{L}_i^c$. Let $\mathbf{S}_i^*$

denote the majorization schedule for this system. The following lemma proves that the schedules $\{\mathbf{S}_i^*\}_{i=1,\ldots,n}$ are feasible transmission schedules for the buffer constrained multicast scenario. This result hinges on a key *property of majorization* schedules (Lemma 3.2).

Lemma 5.1 *The schedule $\mathbf{S}_i^*$ satisfies the following constraints.*

$$\max\{\mathbf{S}^*_{p(i)} - vec(b_{p(i)}), \max_{j\in s(i)} \mathbf{S}_j^*\} \leq \mathbf{S}_i^* \leq \min\{\min_{j\in s(i)} \mathbf{S}_j^* + vec(b_i), \mathbf{S}^*_{p(i)}\}, \quad i = 1, \ldots, n.$$

Proof: It suffices to establish the following,

$$\mathbf{S}^*_{p(i)} - vec(b_{p(i)}) \leq \mathbf{S}_i^* \leq \mathbf{S}^*_{p(i)}, \quad i = 1, \ldots, n. \tag{21}$$

Define $\mathbf{L}_i$ and $\mathbf{U}_i$ as

$$\begin{aligned} \mathbf{L}_i &= \max\{\mathbf{A} - vec(G_{p(i)}), \mathbf{L}_i^c\}, \\ \mathbf{U}_i &= \min\{\mathbf{A}, \mathbf{U}_i^c\}. \end{aligned}$$

Recall that $\mathbf{S}_i^*$ is the majorization schedule associated with $\mathbf{L}_i$ and $\mathbf{U}_i$, $i = 1, \ldots, n$. We have the following inequalities

$$\mathbf{L}_i \leq \mathbf{L}_{p(i)} \leq \mathbf{L}_i + vec(b_{p(i)})$$

and

$$\mathbf{U}_i \leq \mathbf{U}_{p(i)} \leq \mathbf{U}_i + vec(b_{p(i)})$$

which follow from the following properties of max and min, $\max(a, b) + c \geq \max(a + c, b)$ and $\min(a, b) + c \geq \min(a + c, b)$ and the definition of U_i^c. Inequalities (21) follow from these inequalities coupled with an application of Lemma 3.2. ∎

The following lemma is needed to establish the main results of the section. The proof is presented in Appendix A.

Lemma 5.2 *The majorization schedules $\{\mathbf{S}_i^*\}_{i=1}^n$ associated with the finite source single link problems with arrival vector $\mathbf{A}$, source buffers $\{G_i\}$, and buffer overflow and underflow vectors $\{\mathbf{U}_i^c\}_{i\in V}$ and $\{\mathbf{L}_i^c\}_{i\in V}$ satisfy the following relations, for all $i \in V$*

$$\begin{aligned} \max\{\mathbf{A} - G_{p(i)}, \mathbf{L}_i^c\} &\leq \max\{\mathbf{S}_{p(i)} - vec(b_{p(i)}), \max_{j\in s(i)} \mathbf{S}_j\}, \\ \min\{\min_{j\in s(i)} \mathbf{S}_j + vec(b_i), \mathbf{S}_{p(i)}\} &\leq \min\{\mathbf{U}_i^c, \mathbf{A}\} \end{aligned}$$

where it is understood that for $i \in V_l$, $\max_{j\in s(i)} \mathbf{S}_j \equiv \min_{j\in s(i)} \mathbf{S}_j \equiv \mathbf{D}_i$.

We now state and prove the following result regarding whether a feasible set of transmission schedules exists for a given buffer allocation $\{b_i\}_{i\in V}$ and leaf consumption vectors $\{\mathbf{D}_i\}_{i\in V_l}$. The following theorem answers the first question raised in Section 4.2.

Theorem 5.1 *Consider the upper and lower constraints* $\mathbf{U}_i$ *and* $\mathbf{L}_i$ *associated with the finite source single link problems with arrival vector* $\mathbf{A}$*, source buffer* $G_{p(i)}$*, and buffer overflow and underflow vectors* $\mathbf{U}_i^c$ *and* $\mathbf{L}_i^c$*. Then,*

$$\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}) \neq \emptyset \Leftrightarrow \forall i \in V \quad (\mathbf{L}_i^c \leq \mathbf{U}_i^c) \tag{22}$$

Proof:

($\Rightarrow$): Suppose $\mathcal{S}(T, \mathbf{A}, \{b_i\}, \{\mathbf{D}_i\}) \neq \emptyset$. Then there exists a feasible set of schedules $\{\mathbf{S}_i\}_{i=1,\ldots,N}$ which satisfy relations (10) and (11). Then Lemma 5.2 implies that $\forall i \in V \quad (\mathbf{L}_i^c \leq \mathbf{U}_i^c)$.

($\Leftarrow$): $\forall i \in V \quad (\mathbf{L}_i^c \leq \mathbf{U}_i^c)$ implies that a feasible transmission schedule $\mathbf{L}_i \leq \mathbf{S}_i \leq \mathbf{U}_i$ exists, and hence the majorization schedule $\mathbf{S}_i^*$ exists for the single link problem $\forall i$. Now we have shown in Lemma 5.1 that $\{\mathbf{S}_i^*\}$ satisfy the feasibility criteria for the multicast tree. That is $\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\})$ is nonempty.

■

The fact that $\{\mathbf{S}_i^*\}_{i=1}^n$ satisfy the feasibility criteria (Lemma 5.1) together with Lemma 5.2 yield the following theorem regarding the optimality of $\{\mathbf{S}_i^*\}_{i=1}^n$. This answers the second question raised earlier in Section 4.2.

Theorem 5.2 *The majorization schedules* $\{\mathbf{S}_i^*\}_{i=1}^n$ *associated with the finite source single link problems with arrival vector* $\mathbf{A}$*, source buffers* $\{G_i\}_{i\in V}$*, and buffer overflow and underflow vectors* $\{\mathbf{U}_i^c\}_{i\in V}$ *and* $\{\mathbf{L}_i^c\}_{i\in V}$ *satisfy the following relations,*

$$\mathbf{S}_i^* \prec \mathbf{S}_i, \qquad \forall \{\mathbf{S}_i\} \in \mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}_{i\in V_l}).$$

5.1 Buffer Feasibility Check

Based on Theorem 5.1, we now present (see Figure 6) a simple algorithm *Check Feasibility* for checking if the buffer allocation is feasible. This returns True iff $\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}_{i\in V_l}) \neq \emptyset$, otherwise returns False. The algorithm involves traversing the distribution tree a number of times. Each traversal moves either up the tree starting at the leaves (*upward traversal*) or down the tree starting at the root node 0 (*downward traversal*), processing all nodes at the same level before going to the next level :

- (Step 1). This uses relation (17) to compute $\mathbf{U}_i^b$ and $\mathbf{D}_i = \max_{j\in s(i)} \mathbf{D}_j, \forall i \in V \setminus V_l$.
- (Step 2). This uses (18) to compute $\mathbf{U}_i^c$, and (20) to compute $\mathbf{L}_i^c$.
- (Step 3). This checks for feasibility, and uses Theorem 5.1.

Given that $\mathbf{U}_i^c$,and $\mathbf{L}_i^c$ can be computed in $O(N)$ time, the complexity of the above algorithm is $O(nN)$.

PROCEDURE Check_feasibility $(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}_{i \in V_l})$
1. Traverse Up. Compute $\mathbf{U}_i^b$, and $\mathbf{D}_i$, $\forall i \in V$.
2. Traverse Down. Compute $\forall i \in V$, $\mathbf{U}_i^c$, and $\mathbf{L}_i^c$.
3. At each node $i \in V$,
   If $(\mathbf{L}_i^c \leq \mathbf{U}_i^c)$ proceed to next node else return *False*.
   If $(\mathbf{L}_i^c \leq \mathbf{U}_i^c) \forall i \in V_l$ return *True*.
END PROCEDURE

Figure 6: Algorithm *Check Feasibility*

PROCEDURE Optimal_schedule_set $(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}_{i \in V_l})$
1. Traverse Up. Compute $\mathbf{U}_i^b$, and $\mathbf{D}_i$, $\forall i \in V$.
2. Traverse Down. Compute $\forall i \in V$, $\mathbf{U}_i^c$, $\mathbf{L}_i^c$, and $G_{p(i)}$.
3. Traverse Up or Down. Compute $\mathbf{S}_i^*$.
END PROCEDURE

Figure 7: Algorithm *Compute Smooth*

5.2 Optimal Multicast Smoothing Algorithm

We now present a simple algorithm for computing the optimal smoothed transmission schedules for the multicast tree, given a feasible buffer allocation to the nodes in the tree. This involves traversing the distribution tree 3 times using the steps outlined below. The optimal multicast smoothing algorithm *Compute Smooth* is presented in Figure 7. The first 2 steps are identical to that in Figure 6, with the additional computation of $G_{p(i)}$ using $G_{p(i)} = G_{p(p(i))} + b_{p(i)}$. Step 3 computes $\mathbf{S}_i^*$, the majorization schedule associated with the lower and upper constraints $\mathbf{L}_i = \max(\mathbf{A} - vec(G_{p(i)}), \mathbf{L}_i^c)$ and $\mathbf{U}_i = \min(\mathbf{A}, \mathbf{U}_i^c)$.

By Theorem 5.2, the set $\{\mathbf{S}_i^*\}_{i=1}^n$ is optimal. Given that $\mathbf{S}_i^*$ can be computed in $O(N)$ time, the complexity of the above algorithm is $O(nN)$. Note that differential caching at intermediate node buffers plays a crucial role in this optimal smoothing, by temporarily caching differences between, and thereby temporally decoupling to some extent, the transmissions between faster and slower sections of the distribution tree.

6 Rate Constrained Optimal Multicast Smoothing

In this section we develop solutions to the rate constrained multicast problems outlined in Section 4.3. A key step in our approach involves exploiting the results for the single link smoothing problem, in particular, the properties of majorization and lazy transmission schedules.

6.1 Minimum Startup Delay

For each $i \in V \setminus \{0\}$, consider a rate-constrained single link problem with the rate constraint r_i, the arrival vector $\mathbf{A}$ and the client playback vector $\mathbf{D}$. Let b_i^* and w_i^* be the minimum buffer allocation and startup delay required for this single link problem. Then, the minimum common startup delay for the all the clients in the distribution tree is given by $w^* = \max_{k \in V \setminus \{0\}} w_k^*$.

Given this minimum startup delay w^* and assuming that the root server starts video transmission at time 0, the playback vector at client $i \in V_l$ is then $\mathbf{D}(w^*)$.

6.2 Optimal Buffer Allocation

We now proceed to address the *Optimal Buffer Allocation* problem listed before. For this we need some additional concepts. For $i \in V \setminus \{0\}$, define the *effective buffer requirement* b_i^e recursively as follows.

$$b_i^e = \begin{cases} b_i^*, & i \in V_l, \\ \max\{b_i^*, \max_{k \in s(i)} b_k^e\}, & i \in V \setminus V_l. \end{cases} \tag{23}$$

Clearly, $b_i^* \leq b_i^e \leq b_{p(i)}^e$. Note that b_i^e *is the largest buffer allocated to any node in the subtree rooted at node* i. We shall see later that b_i^e is the *minimal* buffer allocation required for the subtree rooted at node i such that a set of feasible schedules exists for the nodes in the tree.

Now for $i \in V \setminus \{0\}$, define

$$\hat{b}_i = \begin{cases} b_i^e, & i \in V_l, \\ b_i^e - \min_{k \in s(i)} b_k^e, & i \in V \setminus V_l. \end{cases} \tag{24}$$

Given this set of buffer allocations $\{\hat{b}_i\}_{i \in V \setminus \{0\}}$, we define the *effective buffer underflow vector* at node i, $\mathbf{L}_i^b$, as $\mathbf{L}_i^b = \mathbf{D}(w^*)$,
and the *effective buffer overflow vector* at node i, $\mathbf{U}_i^b$, as

$$\mathbf{U}_i^b = \begin{cases} \mathbf{D}(w^*) + vec(b_i^e), & i \in V_l, \\ \min_{k \in s(i)} \mathbf{U}_k^b + vec(\hat{b}_i), & i \in V \setminus V_l. \end{cases} \tag{25}$$

Then

Lemma 6.1 *The effective overflow vector has the following property:*

$$\mathbf{U}_i^b \leq \mathbf{U}_{p(i)}^b \leq \mathbf{U}_i^b + vec(\hat{b}_{p(i)}), \tag{26}$$

$$\mathbf{U}_i^b = \mathbf{D}(w^*) + vec(b_i^e). \tag{27}$$

Proof: The relation (26) follows easily from the definitions of $\mathbf{L}_i^b$ and $\mathbf{U}_i^b$. We prove (27) by induction based the height of the tree. The *height*, h, of a node, or of the (sub)tree rooted at the node, is defined as the longest distance from the node to any of its leaf nodes. Leaf nodes have a height of 0.

First consider a node i of height $h = 0$, i.e., $i \in V_l$. In this case $\mathbf{U}_i^b = \mathbf{D}(w^*) + vec(b_i^e)$ follows from the definition of b_i^e.

Suppose that the relation (27) holds for all nodes of height $h \geq 0$. We show that it also holds for any node i of height $h+1$. By definition of $\mathbf{U}_i^b$,

$$\begin{aligned}\mathbf{U}_i^b &= \min_{k\in s(i)} \mathbf{U}_k^b + vec(\hat{b}_i) = \min_{k\in s(i)} \{\mathbf{D}(w^*) + vec(b_k^e)\} + vec(b_i^e - \min_{k\in s(i)} b_k^e) \\ &= \mathbf{D}(w^*) + vec(\min_{k\in s(i)} b_k^e + b_i^e - \min_{k\in s(i)} b_k^e) = \mathbf{D}(w^*) + vec(b_i^e).\end{aligned}$$

■

For $i \in V \setminus \{0\}$, let $\mathbf{S}_i^*$ be the majorization schedule with respect to the lower and upper constraint vectors $(\mathbf{L}_i^b, \min\{\mathbf{A}, \mathbf{U}_i^b\})$. As in the case of the single problem, we show that the set of these majorization schedules, $\{\mathbf{S}_i^*\}_{i\in V}$ (where $\mathbf{S}_0^* \equiv \mathbf{A}$), is a set of feasible schedules for the rate constrained multicast smoothing system. Namely,

Theorem 6.1 *The schedule $\mathbf{S}_i^*$, $i \in V \setminus \{0\}$, satisfies the following constraints.*

$$\max\{\mathbf{S}_{p(i)}^* - vec(\hat{b}_{p(i)}), \max_{j\in s(i)} \mathbf{S}_j^*\} \leq \mathbf{S}_i^* \leq \min\{\min_{j\in s(i)} \mathbf{S}_j^* + vec(\hat{b}_i), \mathbf{S}_{p(i)}^*\}, \tag{28}$$

$$\textit{and} \quad peak(\mathbf{S}_i^*) \leq r_i, \tag{29}$$

where in the above it is understood that $\max_{j\in s(i)} \mathbf{S}_j^* \equiv \mathbf{D}$ for $i \in V_l$ and $\mathbf{S}_0^* = \mathbf{A}$.

Proof: To establish (28), it suffices to show

$$\mathbf{S}_{p(i)}^* - vec(\hat{b}_{p(i)}) \leq \mathbf{S}_i^* \leq \mathbf{S}_{p(i)}^*, \quad i \in V. \tag{30}$$

Let $\mathbf{L}_i = \mathbf{L}_i^b$, and $\mathbf{U}_i = \min(\mathbf{A}, \mathbf{U}_i^b)$. Since $\mathbf{S}_i^*$ is the majorization schedule with respect to $\mathbf{L}_i$ and $\mathbf{U}_i$, from Lemma 6.1 we have $\mathbf{L}_i = \mathbf{L}_{p(i)} < \mathbf{L}_i + vec(\hat{b}_{p(i)})$ and $\mathbf{U}_i \leq \mathbf{U}_{p(i)} \leq \mathbf{U}_i + vec(\hat{b}_{p(i)})$.
Inequalities (30) then follow from these inequalities and an application of Lemma 3.2.

To prove (29), we note that for $i \in V_l$, this follows easily from the definition of $\hat{b}_i$ and Theorem 3.2. We now proceed with the case where $i \in V \setminus V_l$. It suffices to show that $\mathbf{S}_i^*$ is a feasible schedule for the rate-constrained single link smooth problem with the rate constraint r_i, arrival vector $\mathbf{A}$ and the playback vector $\mathbf{D}$. Let $\mathbf{L}_i = \mathbf{L}_i^b$ and $\mathbf{U}_i = \min\{\mathbf{A}, \mathbf{U}_i^b\}$. From Lemma 6.1 and the definition of b_i^e, we have $\mathbf{L}_i = \mathbf{D}(w^*)$ and $\mathbf{U}_i \geq \min\{\mathbf{A}, \mathbf{D}(w^*) + vec(b_i^*)\}$. From the definition of w^*, it is clear that $w^* \geq w_i^*$. Since $\mathbf{S}_i^*$ is the majorization schedule with respect to $(\mathbf{L}_i, \mathbf{U}_i)$, from Lemma 3.2 $\mathbf{S}_i^*$ is majorized by the majorization schedule with respect to $(\mathbf{D}(w^*), \min\{\mathbf{A}, \mathbf{D}(w^*) + vec(b_i^*)\})$. This together with Theorem 3.2 yield (29). ■

PROCEDURE Optimal_buffer (T, $\{r_i\}$, $\mathbf{A}$, $\mathbf{D}$)
1. Traverse Up or down. At each node $i \in V \setminus \{0\}$,
   compute $\mathbf{S}_i^{late}$. Then determine b_i^* and w_i^* .
2. Determine $w^* = \max_{i \in V \setminus \{0\}} w_i^*$.
3. Traverse Up. $\forall i \in V \setminus \{0\}$, determine b_i^e and finally $\hat{b}_i$.
END PROCEDURE

Figure 8: Algorithm *Allocate Buffer*. Optimal buffer allocation for rate constrained multicast of streaming video, given the distribution tree T, rate constraints $\{r_i\}_{\{i \in V \setminus \{0\}\}}$, and consumption vector $\mathbf{D}$ at the leaves of the distribution tree. See text for description of steps.

Remark: We next note the following important result for the rate-constrained multicast scenario. For $i \in V \setminus \{0\}$, define r_i^e recursively as follows. For $i \in V_l$, $r_i^e = r_i$. For $i \in V \setminus V_l$, $r_i^e = \min\{r_i, \min_{j \in s(i)} r_j^e\}$. Then the majorization schedule $\mathbf{S}_i^*$ with respect to $(\mathbf{L}_i^b, \mathbf{U}_i^b)$ is also the majorization schedule for a rate constrained single link problem with the rate constraint r_i^e, the arrival vector $\mathbf{A}$ and the playback vector $\mathbf{D}$. This follows from Theorem 3.2 and the fact that $\mathbf{L}_i^b = \mathbf{D}(w^*)$ and $\mathbf{U}_i^b = \mathbf{D}(w^*) + vec(b_i^e)$ (Lemma 6.1). As a consequence of Theorem 6.1, under the same buffer allocation $\{\hat{b}_i\}_{i \in V \setminus \{0\}}$ and startup delay w^*, *the set of the majorization schedules* $\{\mathbf{S}_i^*\}_{i \in V}$ *gives us the set of the "smoothest" schedules among all feasible schedules for the rate constrained multicast smoothing problem.*

The next theorem is the key to the buffer allocation problem, and establishes the optimality of the buffer allocation $\{\hat{b}_i\}_{i \in V \setminus \{0\}}$. The proof of the theorem is presented in Appendix B.

Theorem 6.2 *The buffer allocation* $\{\hat{b}_i\}_{i \in V \setminus \{0\}}$ *is optimal in the sense that it minimizes, among all the feasible schedules for the system, both the total buffer allocation,* $\sum_{i \in V \setminus \{0\}} \hat{b}_i$, *and the maximal buffer allocated for any node in the subtree rooted at node* i *(namely, the effective buffer allocation* b_i^e *at node* i*),* $i \in V \setminus \{0\}$. *As a result, any smaller total buffer allocation will not result in a feasible set of transmission schedules for the system.*

With the optimality of the buffer allocation $\{\hat{b}_i\}_{i \in V}$ established in Theorem 6.2, we now present a simple algorithm to compute the optimal buffer allocation for a given distribution tree. The algorithm involves 3 traversals through the distribution tree from the leaves, processing all the nodes at the same level before proceeding to the next one. The algorithm *Allocate Buffer* is presented formally in Figure 8. It operates as follows.

- (Step 1). Determine b_i^* and w_i^* using relations (2) and (3) respectively.
- (Step 2). Determine the common minimum startup delay $w^* = \max_{i \in V \setminus \{0\}} w_i^*$.
- (Step 3). At node i, b_i^e is determined using relation (23), and then $\hat{b}_i$ is determined using (24).

Since $\mathbf{S}_i^{late}$ can be computed in time $O(N)$, it is clear that the computation complexity of the above

algorithm is $O(nN)$.

Note that, once the optimal buffer allocation is obtained, we can use the algorithm *Compute Smooth* outlined in Figure 7 to compute the set of optimally smoothed schedules for the multicast tree.

7 Performance Evaluation

In the previous sections, we developed techniques for optimal multicast smoothing and proved the optimality of our approach using properties of the single link smoothing problem.

We next present trace-based evaluations of the the impact of multicast smoothing, and differential caching at intermediate nodes in the distribution tree in reducing network resource requirements for transmitting streaming video. The results can help guide the selection of buffer sizing in a real system, to maximize the benefits of smoothed multicast transmission. In this context, an important metric from an admission control and system provisioning point of view is the total bandwidth $TOTAL$ that needs to be reserved in the tree for supporting this multicast. Let $r(i)$ be the peak rate of the transmission schedule being transmitted to node i from its parent. We assume a simple constant bit rate (CBR) bandwidth reservation model, where the bandwidth along any link is allocated once and is guaranteed for the entire duration of the transmission. Given this CBR reservation, $TOTAL$ is lower bounded by the sum of the peak rates of the transmission schedules being used along each link in the tree, i.e., $TOTAL \geq \sum_{j \in V \setminus \{0\}} r(j)$.

A client (leaf node in the tree) may be charged based on the bandwidth allocation on the path to that client. Two important metrics from a client's perspective, then are :

- allocation along the entire path to the leaf i based on the worst case peak rate on any portion of the path, $SUM\_MAX(i) = (|P(i)| - 1) * (\max_{j \in P(i)} r(j))$.
- the sum of the peak rates of the smoothed transmission schedules along each link on the path from the source to the client $SUM(i) = \sum_{j \in P(i)} r(j)$.

We present trace-driven simulation experiments based on a constant-quality VBR MPEG-2 encoding of a 17-minute segment of the movie *Blues Brothers*. The stream is encoded at 24 frames/second and has a mean rate of 1.48 Mbits/second, peak rate of 44.5 Mbits/second, and an irregular group-of-pictures structure ($IPPPPP\ldots$), with no B frames, and I frames only at scene changes. We consider the buffer constrained multicast problem, and assume that the root of the distribution tree is also the source of the streaming video, i.e, $b_0 = D_N$, and $A_0 = D_N$. We present results for a full 3-ary tree of depth 4. The leaves (clients) have buffers drawn randomly from the set $\{0.512, 1, 2, 4, 8, 16, 32\}$ MB, with only one leaf each having 32 MB and 512 KB. We also assume that all the internal nodes in the tree have identical buffer space (if any).

A *baseline* dissemination algorithm would multicast the unsmoothed video to all the clients. Our algorithm *Compute Smooth* outlined in Section 5 produces optimal transmission schedules along each link, such

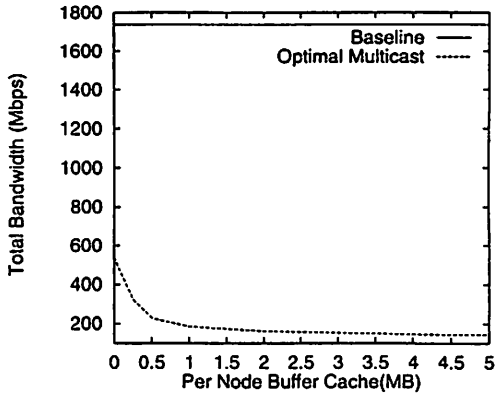
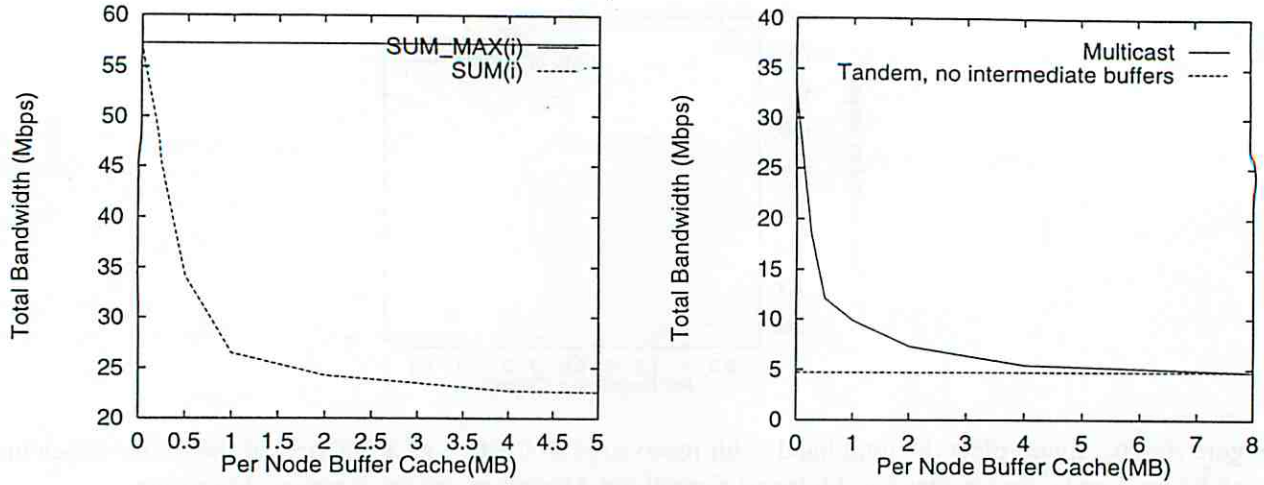


Figure 9: This figure plots the total bandwidth reservation in the tree as a function of the buffer allocation to each internal node, for the Optimal Multicast Smoothing Algorithm and the Baseline Algorithm.

that the peak rate and rate variability along each link is minimized. In Figure 9, we plot $TOTAL$ for the baseline approach and compare it against our algorithm. For the baseline, $TOTAL = 1736$ Mbps. Without using any internal buffering, our approach results in a bandwidth allocation requirement of 541 Mbps, by performing multicast smoothing into the client buffers - a saving of more than a factor of 3. With additional buffering at the internal nodes, the total bandwidth allocation with *Compute Smooth* decreases, initially very rapidly, and then more slowly. For example, with only an additional 512 KB per internal node (i.e., a total additional internal buffering of only 6MB), the total bandwidth allocation reduces by a further factor of 2 beyond the corresponding value for zero internal buffering. This indicates that only a few megabytes of total additional buffer caching in the tree are required to produce significant resource reductions with the optimal multicast smoothing algorithm (*Compute Smooth*).

We next focus on the total bandwidth allocation on the path to the client with the smallest (largest) buffer. In a distribution tree architecture, if leaves have heterogeneous buffer allocations, a smaller leaf buffer can limit how much workahead transmission can be performed on the path to a client with a larger buffer. As a consequence, the bandwidth allocation on the path to the larger client will be higher than if the smaller leaf was absent. The presence of some buffer at an intermediate node (which is a branch point on the path to the 2 nodes) will allow this internal node to temporarily store part of the incoming video. This differential caching may allow more aggressive smoothing to the larger client, by partially decoupling the downstream transmissions to the two leaves.

Figure 10(a) plots $SUM\_MAX(i)$ and $SUM(i)$ along the path to the client with the smallest buffer (512 KB), as a function of the buffer allocation at any internal node . We see that $SUM(i)$ decreases sharply with even a small increase in the buffer allocation to internal nodes. For example, the allocation reduces from 57 Mbps for no internal buffer to 26 Mbps with just 1 MB at each internal buffer cache. This is because a larger buffer at an internal node, present a larger virtual buffer to a smoothing node higher up in the tree, thereby allowing more potential for workahead smoothing higher up in the tree. The graphs indicate that for even such small buffer sizes, there are significant benefits in reserving according to $SUM(i)$ as compared to



(a) Smallest Client (512 KB)

(b) Largest Client (32 MB)

Figure 10: (a) plots $SUM\_MAX(i)$ and $SUM(i)$ for the the client with the smallest buffer (512 KB) (b) plots $SUM(i)$ for for the multicast and tandem scenarios. The startup delay is $w = 12$ frames (0.5 sec).

$SUM\_MAX(i)$. For the same 1 MB internal buffer the bandwidth reservation using $SUM(i)$ is 50% of that for $SUM\_MAX(i)$.

Figure 10(b) plots the total bandwidth allocation ($SUM(i)$) along the path to the client with the largest buffer (32 MB). It also plots $SUM(i)$ assuming that there are no clients with smaller buffers, and no internal buffer caches. This latter case reduces to the tandem smoothing problem with 32 MB client buffer, and no internal buffers. We see that the bandwidth allocation for the client can be much higher for the multicast scenario than for the tandem case. For the above tree, with no buffering at the internal nodes, the bandwidth reservation is 6 times that for tandem, but with only an additional 1 MB buffer per internal node, it reduces to about twice the tandem value. The results indicate that with a few MB internal buffer space, we can get close to the performance achieved in the tandem situation.

8 Conclusions

The multi-timescale burstiness of compressed video makes it a challenging problem to provision network resources for delivering such media. This paper considers the problem of delivering streaming multimedia to multiple heterogeneous clients over an internetwork. We proposed using multicasting of smoothed video and differential caching of the video at intermediate nodes to reduce the network bandwidth requirements of such dissemination. Given a buffer (for differential caching) allocation to the different nodes in the tree, we (i) develop necessary and sufficient conditions for checking if it is possible to transmit the video to all the clients without overflowing any buffer space, and (ii) present an algorithm for computing the set of *optimal* feasible transmission schedules for the tree. In case the multicast tree is rate constrained, we present an algorithm for computing the minimum total buffer allocation to the entire tree, and, the corresponding allocation to each

node, such that *feasible* transmission is possible to all the clients. Initial performance evaluations indicate that there can be substantial benefits from multicast smoothing and differential caching.

In this paper, we have presented a (mostly) analytical and algorithmic treatment of the problem. The next step is to explore actual implementation issues, including designing efficient protocols that will implement the multicast smoothing functionality, by using underlying network support. We also want to extend our current treatment to handle multicast of live video, where the smoothing nodes do not have a priori knowledge of frame sizes. In a related effort, we are investigating a technique for caching the initial frames of popular video streams at an intermediate proxy server [23] for the purpose of increased smoothing benefits and for hiding from the client the effects of a weaker network service model between the server and the proxy. We are looking at these problems, as part of ongoing work.

References

[1] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM*, September 1994.

[2] M. Krunz and S. K. Tripathi, "On the characteristics of VBR MPEG streams," in *Proc. ACM SIGMETRICS*, pp. 192–202, June 1997.

[3] T. V. Lakshman, A. Ortega, and A. R. Reibman, "Variable bit-rate (VBR) video: Tradeoffs and potentials," *Proceedings of the IEEE*, vol. 86, May 1998.

[4] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. Networking*, vol. 6, pp. 397–410, August 1998.

[5] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *Proc. IEEE INFOCOM*, pp. 58–66, April 1997.

[6] J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic, and D. Towsley, "Online smoothing of live, variable-bit-rate video," in *Proc. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 249–257, May 1997.

[7] D. Tennenhouse and D. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 26, April 1996.

[8] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *Proc. ACM SIGCOMM*, September 1998.

[9] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *Proc. ACM Multimedia*, November 1995.

[10] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A network conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. IEEE INFOCOM*, April 1998.

[11] J. Rexford and D. Towsley, "Smoothing Variable-Bit-Rate Video in an Internetwork," *To appear in* IEEE/ACM Trans. Networking, 1999.

[12] G. Sanjay and S. V. Raghavan, "Fast techniques for the optimal smoothing of stored video." To appear in *ACM Multimedia Systems Journal*.

[13] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems*, vol. 8, pp. 85–110, May 1990.

[14] K. Almeroth and M. Ammar, "On the use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communication*, vol. 14, August 1996.

[15] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley, "Online Smoothing of Variable-Bit-Rate Streaming Video," Tech. Rep. 98-75, Department of Computer Science, University of Massachusetts Amherst, 1998.

[16] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Proc. ACM SIGMETRICS*, pp. 222–231, May 1996.

[17] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.

[18] G. Sanjay, "Work-ahead smoothing of video traffic for interactive multimedia applications." Bachelor of Technology Project Report, Dept. of Computer Science and Engineering, Indian Institute of Technology, Madras, May 1997.

[19] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," in *Proc. IS&T/SPIE Multimedia Networking and Computing*, pp. 316–327, February 1997.

[20] S. Sahu, Z.-L. Zhang, J. Kurose, and D. Towsley, "On the efficient retrieval of VBR video in a multimedia server," in *Proc. IEEE Conference on Multimedia Computing and Systems*, pp. 46–53, June 1997.

[21] J. K. Dey, S. Sen, J. Kurose, D. Towsley, and J. Salehi, "Playback restart in interactive streaming video applications," in *Proc. IEEE Conference on Multimedia Computing and Systems*, pp. 458–465, June 1997.

[22] J. Rexford and D. Towsley, "Smoothing Variable-Bit-Rate Video in an Internetwork," in *Proc. SPIE Symposium on Voice, Video, and Data Communications: Multimedia Networks: Security, Displays, Terminals, and Gateways*, November 1997.

[23] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," in *Proc. IEEE INFOCOM*, April 1999.

Appendix

A Proof for Lemma 5.2

Proof of Lemma 5.2: This is accomplished by establishing the following four inequalities,

$$\mathbf{A} - G_{p(i)} \leq \mathbf{S}_{p(i)} - vec(b_{p(i)}) \tag{31}$$

$$L^c_{i,k} \leq \max\{S_{p(i),k} - b_{p(i)}, \max_{j \in s(i)} S_{j,k}\}, \quad k = 1, \ldots, N \tag{32}$$

$$\min\{\min_{j \in s(i)} S_{j,k} + b_i, S_{p(i),k}\} \leq U^c_{i,k}, \quad k = 1, \ldots, N \tag{33}$$

$$\mathbf{S}_{p(i)} \leq \mathbf{A} \tag{34}$$

Inequalities (31) and (34) follow from the definition of $\mathbf{S}_0 = \mathbf{A}$ coupled with successive applications of (12).

Consider inequality (32) and a fixed value of k, $k = 1, \ldots, N$. We begin by ordering the nodes so that $D_{1,k} \geq D_{2,k} \geq \cdots \geq D_{n+1,k}$. In the case that $D_{i,k} = D_{i+1,k}$, we adopt the convention that $l(i) \geq l(i+1)$ where $l(j)$ is taken to be the distance between the root and node j. We proceed by induction on i.

Basis step. Node 1 must be a leaf as a consequence of the ordering convention. It is easily verified that $L^c_{1,k} = D_{1,k}$ and inequality (32) follows directly.

Inductive step. Assume that (32) holds for nodes $1, \ldots, i-1$. We establish it for node i. There are two cases

Case (i) $L^c_{p(i),k} - b_{p(i)} \leq D_{i,k}$: In this case $L^c_{i,k} = D_{i,k}$ and there exists a $j \in s(i)$, $j < i$ such that $L^c_{j,k} = D_{j,k} = L^c_{i,k} = D_{i,k}$ By induction we know that

$$L^c_{j,k} \leq \max\{S_{i,k} - b_i, \max_{l \in s(j)} S_{l,k}\}$$

If $L^c_{j,k} \leq S_{i,k} - b_i$, then $L^c_{i,k} = L^c_{j,k} \leq S_{i,k} - b_i \leq S_{j,k}$ by (12) which establishes (32). If $L^c_{j,k} \leq S_{l,k}$ for some $l \in s(j)$, then $L^c_{i,k} = L^c_{j,k} \leq S_{l,k} \leq S_{j,k}$ by (12) which again establishes (32).

Case (ii) $L^c_{p(i),k} - b_{p(i)} > D_{i,k}$: In this case, it is easy to show that $D_{p(i),k} > D_{i,k}$ and by that by the inductive hypothesis,

$$L^c_{p(i),k} \leq \max\{S_{p(p(i)),k} - b_{p(p(i))}, \max_{j \in s(p(i))} S_{j,k}\}$$

If $L^c_{p(i),k} \leq S_{p(p(i),k} - b_{p(p(i))}$, then $L^c_{i,k} = L^c_{p(i),k} - b_{p(i)} \leq S_{p(p(i)),k} - b_{p(p(i))} - b_{p(i)} \leq S_{p(i),k} - b_{p(i)}$ by (12) which establishes (32). If $L^c_{p(i),k} \leq S_{j,k})$ for some $j \in s(p(i))$, then $L^c_{i,k} = L^c_{p(i),k} - b_{p(i)} \leq S_{j,k} - b_{p(i)} \leq S_{p(i),k} - b_{p(i)} \leq S_{i,k}$ where the last two inequalities follow from (12) thus establishing (32).

This completes the inductive step and the proof of (32).

Now consider inequality (33) and fix k. Assume for now that $b_i > 0$. We again order the nodes, other than the root node, so that $U^c_{1,k} \leq U^c_{2,k} \leq \cdots \leq U^c_{n,k}$. In the case that $U^c_{i,k} = U^c_{i+1,k}$, we adopt the convention that $l(i) \leq l(i+1)$ where $l(j)$ is taken to be the distance between the root and node j. We proceed by induction on i.

Basis step. As a consequence of the ordering convention and the fact that $b_j > 0$ for $j = 1, \ldots, n$, node 1 must be a leaf. Hence, $U^c_{1,k} = D_{1,k} + b_1$ and inequality (33) follows from (12).

Inductive step. Assume that (33) holds for nodes $1, \ldots, i-1$. We establish it for node i. There are two cases

Case (i) $U^c_{i,k} < U^c_{p(i),k}$: In this case $U^c_{i,k} = U^b_{i,k}$ and there exists a $j \in s(i)$, $j < i$ such that $U^c_{j,k} = U^b_{j,k} = U^c_{i,k} - b_i = U^b_{i,k} - b_i$. By induction we know that

$$\min\{\min_{l \in s(j)} S_{l,k} + b_j, S_{i,k}\} \leq U^c_{j,k}$$

If $S_{i,k} \leq U^c_{j,k}$ then $U^c_{i,k} = U^c_{j,k} + b_i \geq S_{i,k} + b_i \geq S_{j,k} + b_i$ where the second inequality follows from (12). If $S_{l,k} + b_j \leq U^c_{j,k}$ for some $l \in s(j)$, then $U^c_{i,k} = U^c_{j,k} + b_i \geq S_{l,k} + b_j + b_i \geq S_{j,k} + b_i$ where again the second inequality follows from (12).

Case (ii) $U^c_{i,k} = U^c_{p(i),k}$: In this case, according to the ordering convention and the inductive hypothesis,

$$\min\{\min_{j \in s(p(i))} S_{j,k} + b_{p(i)}, S_{p(p(i)),k}\} \leq U^c_{p(i),k}$$

If $S_{p(p(i)),k} \leq U^c_{p(i),k}$, then $U^c_{i,k} = U^c_{p(i),k} \geq S_{p(p(i)),k} \geq S_{p(i),k}$ where the last inequality follows from (12). If $S_{j,k} + b_{p(i)} \leq U^c_{p(i),k}$ for some $j \in s(p(i))$ then $U^c_{i,k} = U^c_{p(i),k} \geq S_{j,k} + b_{p(i)} \geq S_{p(i),k}$ where the last inequality follows from (12).

This completes the inductive step. ∎

B Proof for Theorem 6.2

Proof of Theorem 6.2: We prove by induction on the height of the distribution tree.

For any node i of height $h = 0$, i.e., $i \in V_l$. It is clear that $\hat{b}_i = b_i^e = b_i^*$ is the minimum buffer required for the existence of a feasible transmission schedule $\mathbf{S}_i$ such that $peak(\mathbf{S}_i) \leq r_i$.

Suppose the theorem holds for any subtree rooted at any node of height $h-1$, $h \geq 1$. We show that it is also true for any subtree rooted at any node of height h. Consider such a node i of height h. We want to prove that the total buffer allocation at the subtree rooted at node i as well as the maximal buffer allocated to any node in the subtree rooted at node i (the effective buffer allocation at node i, b_i^e) is minimized by our choice of $\hat{b}_i$.

We first show that $\hat{b}_i$ is the minimal feasible buffer allocation at node i. Let $|s(i)| = m$. Without loss of generality, we label the child nodes of i as $1, \ldots, m$, where $b_1^e \leq \cdots \leq b_m^e$. Let b_i^{min} denote the minimum buffer allocation at node i satisfying (15) and (16), given that the buffer allocation at its children is $\{\hat{b}_j\}_{j \in s(i)}$. From Lemma 3.5, the Remark after Theorem 6.1 and, the optimality of buffer allocation $\{b_k^e, k \in s(i)\}$, we have $b_i^{min} \geq \max_{1 \leq k < l \leq m}\{\max\{\mathbf{S}_l^* - \mathbf{S}_k^*\}\}$ and $b_i^{min} \geq \max_{1 \leq k \leq m}\{\max\{\mathbf{S}_i^* - \mathbf{S}_k^*\}\}$. Under the assumption that $b_1^e \leq \cdots \leq b_m^e$, we have $\mathbf{S}_1^* \leq \cdots \leq \mathbf{S}_m^*$. This follows from Lemma 3.2 together with the fact that $\mathbf{L}_i^b = \mathbf{D}(w^*)$ and $\mathbf{U}_i^b = \mathbf{D}(w^*) + vec(b_i^e)$ (Lemma 6.1). Therefore, $b_i^{min} \geq \max\{\mathbf{S}_m^* - \mathbf{S}_1^*, \mathbf{S}_i^* - \mathbf{S}_1^*\} = \max\{b_m^e, b_i^e\} - b_1^e = b_i^e - b_1^e = \hat{b}_i$, where the last two equalities follow from the definition of b_i^e and $\hat{b}_i$. From Theorem 6.1, $\hat{b}_i$ is a feasible buffer allocation at node i. Thus $b_i^{min} = \hat{b}_i$. Hence we establish that our choice of buffer $\hat{b}_i$ for node i is the minimum feasible buffer allocation. Given the optimality of buffer allocation $\{b_k^e, k \in s(i)\}$ at its child nodes, we see that $b_i^e = \hat{b}_i + \min_{k \in s(i)} b_k^e$ is also the minimal feasible effective buffer allocation at node i.

We now demonstrate that increasing the buffer allocation at any child node $k \in s(i)$ does not reduce the effective buffer requirement at node i, b_i^e, nor the total buffer allocation to the subtree rooted at node i.

Suppose that the effective buffer allocation b_k^e, $1 \leq k \leq m$, is increased by x amount. From the proof of Theorem 3.2, we see that there exists t such that $S_{k,t}^* = U_{k,t} = D_t(w^*) + b_k^e$ (the last equality follows from Lemma 6.1). Any feasible schedule $\mathbf{S}_k(x)$ with the increased buffer capacity at node k must satisfy the condition that $S_{k,t}(x) \leq D_t(w^*) + b_k^e + x$. Thus $\max\{\mathbf{S}_m^* - \mathbf{S}_k(x)\} \geq b_m^e - x - b_k^e$ and $\max\{\mathbf{S}_i^* - \mathbf{S}_k(x)\} \geq b_i^e - x - b_k^e$. Therefore, the buffer $\hat{b}_i$ at node i can be decreased by at most x amount. However, the total buffer allocation to the subtree rooted at node i does not decrease, since the buffer allocation at node k has increased by x units. Moreover, b_i^e will never decrease. This concludes the proof of the theorem. ∎

C Buffer Constrained Optimal Multicast Smoothing

A special case of buffer constrained multicast smoothing is when all the clients have the same playback vector, i.e., $\mathbf{D}_i = \mathbf{D}_j = \mathbf{D}, i, j \in V_l, i \neq j$. The proof for Lemma 5.2 then becomes

Proof of Lemma 5.2: This is accomplished by establishing the following four inequalities,

$$\mathbf{A} - G_{p(i)} \leq \mathbf{S}_{p(i)} - vec(b_{p(i)}) \tag{35}$$

$$L^c_{i,k} \leq \max\{S_{p(i),k} - b_{p(i)}, \max_{j \in s(i)} S_{j,k}\}, \quad k = 1, \ldots, N \tag{36}$$

$$\min\{\min_{j \in s(i)} S_{j,k} + b_i, S_{p(i),k}\} \leq U^c_{i,k}, \quad k = 1, \ldots, N \tag{37}$$

$$\mathbf{S}_{p(i)} \leq \mathbf{A} \tag{38}$$

Inequalities (35) and (38) follow from the definition of $\mathbf{S}_0 = \mathbf{A}$ coupled with successive applications of (12).

Consider inequality (36). and a fixed value of k $k = 1, \ldots, N$. We begin by ordering the nodes so that $L^c_{1,k} \geq L^c_{2,k} \geq \cdots \geq L^c_{n+1,k}$. In the case that $L^c_{i,k} = L^c_{i+1,k}$, we adopt the convention that $l(i) \geq l(i+1)$ where $l(j)$ is taken to be the distance between the root and node j. We proceed by induction on i.

Basis step. Node 1 must be a leaf as a consequence of the ordering convention. It is easily verified that $L^c_{1,k} = D_k$ and inequality (36) follows directly.

Inductive step. Assume that (36) holds for nodes $1, \ldots, i-1$. We establish it for node i. There are two cases

Case (i) $L^c_{p(i),k} - b_{p(i)} \leq D_k$: In this case $L^c_{i,k} = D_k$ and there exists a $j \in s(i)$, $j < i$ such that $L^c_{j,k} = D_k = L^c_{i,k}$ By induction we know that $L^c_{j,k} \leq \max\{S_{i,k} - b_i, \max_{l \in s(j)} S_{l,k}\}$. If $L^c_{j,k} \leq S_{i,k} - b_i$, then $L^c_{i,k} = L^c_{j,k} \leq S_{i,k} - b_i \leq S_{j,k}$ by (12) which establishes (36). If $L^c_{j,k} \leq S_{l,k}$ for some $l \in s(j)$, then $L^c_{i,k} = L^c_{j,k} \leq S_{l,k} \leq S_{j,k}$ by (12) which again establishes (36).

Case (ii) $L^c_{p(i),k} - b_{p(i)} > D_k$: By the inductive hypothesis, $L^c_{p(i),k} \leq \max\{S_{p(p(i)),k} - b_{p(p(i))}, \max_{j \in s(p(i))} S_{j,k}\}$. If $L^c_{p(i),k} \leq S_{p(p(i),k} - b_{p(p(i))}$, then $L^c_{i,k} = L^c_{p(i),k} - b_{p(i)} \leq S_{p(p(i)),k} - b_{p(p(i))} - b_{p(i)} \leq S_{p(i),k} - b_{p(i)}$ by (12) which establishes (36). If $L^c_{p(i),k} \leq S_{j,k})$ for some $j \in s(p(i))$, then $L^c_{i,k} = L^c_{p(i),k} - b_{p(i)} \leq S_{j,k} - b_{p(i)} \leq S_{p(i),k} - b_{p(i)} \leq S_{i,k}$ where the last two inequalities follow from (12), thus establishing (36).

This completes the inductive step and the proof of (36).

Now consider inequality (37) and fix k. Assume for now that $b_i > 0$. We again order the nodes, other than the root node, so that $U^c_{1,k} \leq U^c_{2,k} \leq \cdots \leq U^c_{n,k}$. In the case that $U^c_{i,k} = U^c_{i+1,k}$, we adopt the convention that $l(i) \leq l(i+1)$ where $l(j)$ is taken to be the distance between the root and node j. We proceed by induction on i.

Basis step. As a consequence of the ordering convention and the fact that $b_j > 0$ for $j = 1, \ldots, n$, node 1 must

be a leaf. Hence, $U^c_{1,k} = D_k + b_1$ and inequality (37) follows from (12).

Inductive step. Assume that (37) holds for nodes $1, \ldots, i-1$. We establish it for node i. There are two cases

Case (i) $U^c_{i,k} < U^c_{p(i),k}$**:** In this case $U^c_{i,k} = U^b_{i,k}$ and there exists a $j \in s(i)$, $j < i$ such that $U^c_{j,k} = U^b_{j,k} = U^c_{i,k} - b_i = U^b_{i,k} - b_i$. By induction we know that $\min\{\min_{l \in s(j)} S_{l,k} + b_j, S_{i,k}\} \leq U^c_{j,k}$. If $S_{i,k} \leq U^c_{j,k}$ then $U^c_{i,k} = U^c_{j,k} + b_i \geq S_{i,k} + b_i \geq S_{j,k} + b_i$ where the second inequality follows from (12). If $S_{l,k} + b_j \leq U^c_{j,k}$ for some $l \in s(j)$, then $U^c_{i,k} = U^c_{j,k} + b_i \geq S_{l,k} + b_j + b_i \geq S_{j,k} + b_i$ where again the second inequality follows from (12).

Case (ii) $U^c_{i,k} = U^c_{p(i),k}$**:** In this case, according to the ordering convention and the inductive hypothesis, $\min\{\min_{j \in s(p(i))} S_{j,k} + b_{p(i)}, S_{p(p(i)),k}\} \leq U^c_{p(i),k}$.
If $S_{p(p(i)),k} \leq U^c_{p(i),k}$, then $U^c_{i,k} = U^c_{p(i),k} \geq S_{p(p(i)),k} \geq S_{p(i),k}$ where the last inequality follows from (12). If $S_{j,k} + b_{p(i)} \leq U^c_{p(i),k}$ for some $j \in s(p(i))$ then $U^c_{i,k} = U^c_{p(i),k} \geq S_{j,k} + b_{p(i)} \geq S_{p(i),k}$ where the last inequality follows from (12).

This completes the inductive step. ∎

For this special case, an alternative to Theorem 5.1 is:

Theorem C.1 *Consider the upper and lower constraints* $\mathbf{U}_i$ *and* $\mathbf{L}_i$ *associated with the finite source single link problems with arrival vector* $\mathbf{A}$*, source buffer* $G_{p(i)}$*, and buffer overflow and underflow vectors* $\mathbf{U}^c_i$ *and* $\mathbf{L}^c_i$*. Then,*

$$\mathcal{S}(T, \{b_i\}, \mathbf{A}, \{\mathbf{D}_i\}) \neq \emptyset \Leftrightarrow \forall i \in V_l \quad (\mathbf{L}^c_i \leq \mathbf{U}^c_i) \tag{39}$$

A consequence of this is that algorithm *Check Feasibility* becomes simpler, as in Step 3, we need to check whether $(\mathbf{L}^c_i \leq \mathbf{U}^c_i)$ only for all leaf nodes $i \in V_l$.