Scalable On-Demand Streaming of Non-Linear Media

Yanping Zhao Derek Eager
Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5A9, Canada
zhao,eager@cs.usask.ca

Mary K. Vernon
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706, USA
vernon@cs.wisc.edu

***Abstract*— A conventional video file contains a single temporally-ordered sequence of video frames. Clients requesting on-demand streaming of such a file receive (all or intervals of) the same content. For popular files that receive many requests during a file playback time, scalable streaming protocols based on multicast or broadcast have been devised. Such protocols require server and network bandwidth that grow much slower than linearly with the file request rate.**

This paper considers "non-linear" video content in which there are parallel sequences of frames. Clients dynamically select which branch of the video they wish to follow, sufficiently ahead of each branch point so as to allow the video to be delivered without jitter. An example might be "choose-your-own-ending" movies. With traditional scalable delivery architectures such as movie theaters or TV broadcasting, such personalization of the delivered video content is very difficult or impossible. It becomes feasible, in principle at least, when the video is streamed to individual clients over a network. This paper analyzes the minimal server bandwidth requirements, and proposes and evaluates practical scalable delivery protocols, for on-demand streaming of non-linear media.

I. Introduction

A conventional video file contains a single temporally-ordered sequence of video frames. Clients that request the same file receive encodings of (all or intervals of) the same frames. We hypothesize here that generalizing this structure to that of a tree or graph, so as to allow parallel sequences of frames among which clients dynamically select during playback, may enable new streaming media applications, as well as enrich existing ones. An example is "choose-your-own-ending" entertainment videos, analogous to the many choose-your-own-ending children's books.

For conventional stored video, a number of scalable streaming protocols based on (IP or application level) multicast or broadcast have been developed. Such protocols require server and network bandwidth that grow much slower than linearly with the file request rate. These include immediate service protocols such as patching [3], [7], [9] and hierarchical stream merging [5], as well as periodic broadcast protocols [1], [6], [8], [10]–[12], [16]. In the immediate service protocols, a new stream is allocated for each incoming client request, and streams serving closely spaced requests for the same file are dynamically "merged" by having clients also listen to one or more earlier streams to receive and buffer data that they will need to play back in the future. In periodic broadcast protocols, the video file is segmented, and each segment is repeatedly broadcast/multicast on one of a number of channels (e.g., IP multicast groups) according to some protocol-dependent transmission schedule. Unlike with the immediate service protocols, clients must wait to begin playback, with the length of the waiting period dependent on the duration of a transmission of the initial segment. For "whole file" playback requests, the best of the immediate service protocols use server bandwidth that grows logarithmically with the file request rate, while the best of the periodic broadcast protocols have start-up delay that decreases exponentially with the (fixed) server bandwidth allotted to the file.

This paper first explores the potential bandwidth savings from using scalable, multicast-based streaming techniques for on-demand delivery of non-linear stored video. As the diversity in the data each client receives increases, the potential benefits of multicast delivery can be expected to diminish. A basic question is whether, or under what conditions, the potential benefits become negligible in this context. This question is addressed through the development of tight lower bounds on the server bandwidth required to support a given file request rate and client start-up delay, for non-linear media files with varying path diversity. Our results indicate that the potential bandwidth savings can be substantial, even for videos with high path diversity.

Scalable streaming protocols achieve bandwidth reductions by transmitting video file data to multiple clients. For the shared transmissions to be possible, at least some clients receive data ahead of when it is needed for playback, buffering it in memory or on disk until its playback point. With non-linear video, however, transmitting data ahead of when it is needed is complicated by uncertainty regarding which branch a client will follow at each branch point. There is a tradeoff between receiving data that the client might not need, and the server bandwidth reduction arising from receiving (needed) data ahead of its playback point, so as to be able to share the transmission with other clients. We investigate various points in this tradeoff using tight lower bounds on the server band-

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada, and by the National Science Foundation under grants ANI-0117810 and EIA-0127857.

To appear in *Proc. IEEE Infocom 2004*, Hong Kong, March 2004.

width required for various classes of protocols. Some of the protocol classes considered make use of advance knowledge of which branch a client will likely follow at each branch point. We consider both the use of measured (over all clients) branch choice frequencies, and client-specific information, as might result from pre-declaration of intended client paths or from client classification.

Our results show that fairly precise *a priori* information regarding client path selection can dramatically reduce server bandwidth requirements as well as the client data overhead of receiving data that is never used. In the absence of such information, strategies that restrict what data clients will receive in advance of knowing whether or not it will be needed, based on how far ahead that data is in the video file rather than more approximate client path predictions, can greatly reduce the client data overhead at relatively small bandwidth cost.

Finally, using insights derived from the bounds we design new immediate service and periodic broadcast protocols for non-linear video, and evaluate the bandwidth savings that they provide. Within each class of protocols, variants are developed that assume the extremes of either no *a priori* path knowledge, or full knowledge. In general, as with our lower bounds, precise *a priori* information regarding client path selection can substantively reduce the server bandwidth requirements.

The remainder of the paper is organized as follows. Section II describes models for non-linear media. Tight lower bounds on the server bandwidth required for a given file request rate and client start-up delay, and the corresponding client data overhead if no *a priori* client path selection information is available, are derived in Section III. Section IV derives the server bandwidth bounds and associated client data overheads for various policies that restrict the data that clients receive ahead of when it is needed. Section V presents new stream merging and periodic broadcast protocols, and comparative performance results. Conclusions are given in Section VI.

II. Non-Linear Media Models

A. Non-Linear Media Structures

The simplest interesting structure for non-linear video is that of a height one tree with root node corresponding to a common initial portion, and child nodes corresponding to multiple possible ending portions. In a "complete path" playback of the video, the client plays the common portion plus one of the ending portions. If the desired variant of the ending portion is chosen sufficiently ahead of the end of the common initial portion (the branch point), the complete path can be played without jitter. In the following, except when stated otherwise, it is assumed that clients make navigation decisions soon enough to avoid jitter, but sufficiently close to the respective branch point that the gap can be neglected in our analysis.

A more general structure is an arbitrary tree, where each node corresponds to a portion of the video, and child nodes correspond to variant subsequent portions. A complete path playback would consist of the common root portion, plus all other portions on a path up to and including a leaf node. This structure can be further generalized to a directed acyclic graph (i.e., paths can converge at shared portions), or a general graph structure. In the latter case, the notion of a "complete path" playback may have no meaning; clients simply start playback at some client-selected video portion and the graph links determine the possible subsequent portions.

The bounds in Sections III and IV are developed for tree structures, although the analysis can be generalized. The immediate service protocols developed in Section V.A are applicable to non-linear media having a general graph structure, while our periodic broadcast protocols in Section V.B are applicable to directed acyclic graphs in which the path lengths to any video portion with multiple parents are identical, and to general tree structures. For clarity, however, we present numerical results only for balanced binary trees in which all video portions have identical playback time, and assuming that each client request is for a complete path playback.

We assume constant bit rate video. Generalizations for variable bit rate video can be developed using similar approaches as for linear media [13], [14], [18].

B. Client Branch Selections

A key issue concerns the relative frequencies with which clients select among alternative portions of the video at branch points. In the context of balanced binary tree structures, we have explored several alternative popularity models. The model for which numerical results will be presented assigns selection probabilities to leaves according to a Zipf distribution, as follows. First, the leaf that will be the most popular is chosen randomly, and assigned the corresponding probability. Then, out of the remaining leaves, a second most popular is chosen randomly, and so on. Once all of the leaves have been given selection probabilities, selection probabilities for all interior video portions can be computed by working up from the leaves.

Other models that were considered include a model in which the leaves are assigned Zipf-distributed selection probabilities in order, with the leftmost leaf the most popular and the rightmost the least popular, and a model in which the selection probabilities at each branch point are Zipf-distributed (specifically, for a branch point with two branches, one branch is selected with probability 2/3, and the other with probability 1/3). Although these other models would appear to differ significantly from the chosen model (in particular, they give more skewed selection probabilities at the branch points near the root of the tree and less skewed probabilities at those near the leaves), they were found to yield very similar results.

C. An Example

Fig. 1 shows a sample non-linear video file structure. Each portion of the video is denoted by a line segment, with branch points denoted by solid black circles. As a tree structure with nodes representing portions of the video, the structure in Fig. 1 corresponds to a balanced binary tree of height 3. In the figure each video portion is labelled by its selection probability, as computed by choosing leaf selection probabilities according to a Zipf distribution, and then working up the tree. Also shown

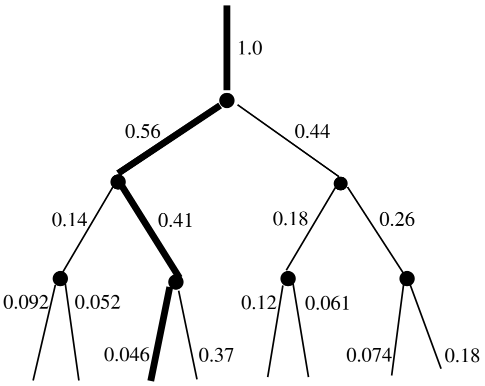


Fig. 1. Example of a Non-Linear Media Structure

TABLE I
NOTATION FOR TREE-STRUCTURED NON-LINEAR MEDIA

Symbol	Definition
V	number of portions of the video file
T	complete path playback time
T_i	playback time of i^{th} portion (root numbered as portion 1)
t_i	i^{th} portion relative start time ($t_1 = 0$)
p_i	probability the selected path includes portion i
α	parameter of Zipf distribution (popularity of j'th most popular item $\propto 1/j^{\alpha}$)
λ	client request rate
λ_i	request rate for i^{th} portion ($\lambda_i = p_i\lambda$)
N	average number of client requests during a playback time ($N = \lambda T$)
N_i	average number of client requests for portion i during time T_i ($N_i = \lambda_i T_i$)
d	maximum client start-up delay
B_{min}	required server bandwidth lower bound, in units of the playback data rate

is the path selected by a particular client, who made the most popular selection at the first branch point (followed in 56% of all client playbacks), and who chose a complete path that is selected in 4.6% of all client playbacks.

D. Server Knowledge of Client Preferences

Of interest are three cases: (1) no *a priori* knowledge is available of the likely path through the video that a particular client will take, (2) only the overall average selection probabilities are known, and (3) more accurate client-specific path prediction is possible, as when the previous behavior of clients is measured, either individually or in aggregate according to some client classification. In the second case, the system might predict that the client will choose the most popular branch at each branch point, in which case the client's choice is correctly predicted with probability equal to the (conditional) selection frequency of the most popular branch. In the third case, we consider in Section IV.B a simple model of client-specific path prediction accuracy in which sufficiently popular branch choices are always successfully predicted, and the other, unpopular branch choices are never predicted. This analytically tractable model has the key advantage, for binary tree structures, of covering a spectrum from path prediction in which only choices of the most popular branch at each branch point are successfully predicted (i.e., the same as if only overall average selection probabilities are employed), to fully accurate prediction in which all branch choices are successfully predicted, depending on the quantification of "sufficiently popular". When an incorrect prediction is made, it is assumed that the prediction is for each of the paths that could have been predicted with probability proportional to its relative popularity.

III. POTENTIAL FOR SCALABLE DELIVERY

With unicast delivery, server and network bandwidth requirements for on-demand streaming are linear in the client request rate. This section analyzes the extent to which server bandwidth requirements might be reduced through use of multicast-based protocols in the context of non-linear media, and the associated client data overheads. Section III.A defines these performance metrics and outlines the analysis approach. In Section III.B, a tight lower bound on the server bandwidth requirement is derived. Section III.C derives the client data overhead required to achieve the server bandwidth bound when no *a priori* information is available regarding client path selection. Classes of policies that restrict the client data overhead are considered in Section IV.

A. Metrics and Analysis Approach

The primary performance metric that is considered is the average server bandwidth used for "complete path" playbacks of a single video file, for given client start-up delay and request rate. Our analysis can be extended to network bandwidth in a similar fashion as for linear media [19]. Also of interest is the average client data overhead, defined as the average amount of data a client receives from video portions on different paths than that taken by the client, and therefore not used, expressed in units of the amount of video data on a complete path.

Using the notation defined in Table I, our lower bound analysis follows the same basic approach as has been used previously for linear media [2], [5], [6], [15]. For a linear media file, and an arbitrary client request that arrives at time t, the file data at each play position x must be delivered no later than time $t+d+x$. If this data is multicast at time $t+d+x$, then (at best) those clients that request the file between time t and $t+d+x$ can receive the same multicast. Assuming Poisson arrivals, the average time from $t+d+x$ until the next request for the file is $1/\lambda$. Therefore, the minimum frequency of multicasts of the data at time offset x is $1/(d+x+1/\lambda)$, which yields a bound on required server bandwidth, in units of the playback data rate, of

$$B_{min}^{linear} = \int_0^T \frac{\mathrm{dx}}{d+x+\frac{1}{\lambda}} = \ln\left(\frac{N}{N\frac{d}{T}+1}+1\right). \quad (1)$$

This bound can be generalized to a broad class of non-Poisson arrival processes, yielding a similar result with difference bounded by a constant [5]. Bounds for non-linear media are derived below by applying similar analysis.

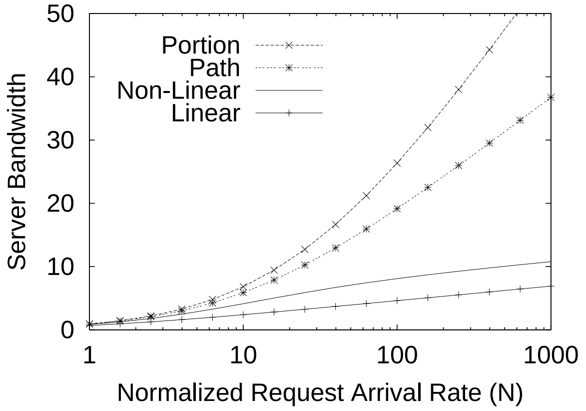


Fig. 2. Server Bandwidth for Non-Linear Media (balanced binary tree with height 3, α = 1, d = 0)

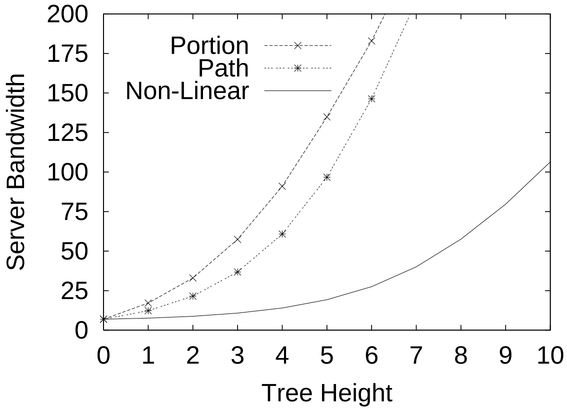


Fig. 3. Impact of Tree Height (α = 1, N = 1000, d = 0)

B. Minimum Required Server Bandwidth

Server bandwidth is minimized when a client listens to every multicast of data that it may need in the future. Note that without *a priori* knowledge of client path selection, this requires that the client listen to any multicast of data in the subtree below its current play point, implying possibly large client data overhead. With perfect *a priori* knowledge of client path selection, the client listens only to all multicasts of data that it will actually use in the future. In either case, noting that the file data at a position x within a video portion i is at (overall) play position $t_i + x$, the above analysis approach yields the tight lower bound

$$\begin{aligned} B_{min}^{non-linear} &= \sum_{i=1}^{V} \int_0^{T_i} \frac{\mathsf{dx}}{d + t_i + x + \frac{1}{\lambda_i}} \\ &= \sum_{i=1}^{V} \ln\left(\frac{N_i}{N_i \frac{d+t_i}{T_i} + 1} + 1 \right). \end{aligned} \tag{2}$$

Fig. 2 shows this bound as a function of the normalized request arrival rate N, for immediate service (d = 0) and for a non-linear media file with a balanced binary tree structure of height 3 and Zipf-distributed leaf selection probabilities as described in Section II with Zipf distribution parameter α = 1. (Alternative random assignments to leaves of the Zipf selection probabilities yield very similar results.)

For comparison purposes, the figure also shows the bound for linear media from eq. 1, and bounds for two approaches in which delivery techniques for linear media are applied to non-linear media. In one of these (*portion*), each portion of the non-linear media file is treated as a separate linear media file, yielding a tight lower bound on required server bandwidth of

$$B_{min}^{portion} = \sum_{i=1}^{V} \int_0^{T_i} \frac{\mathsf{dx}}{d_i + x + \frac{1}{\lambda_i}} = \sum_{i=1}^{V} \ln\left(\frac{N_i}{N_i \frac{d_i}{T_i} + 1} + 1 \right). \tag{3}$$

Here $d_1 = d$, and the terms d_i for $i > 1$ admit the possibility that with this approach, a client selection of video portion i would be required to be made time d_i prior to the end of its parent portion (or, alternatively, that there would be interruption in playback of duration d_i). For the results in the figure it is assumed that $d_i = 0$ for all i. In the other approach (*path*), the client path selection is required to be known *a priori*. Video data is replicated so that each complete path through the tree structure can be stored as a separate file. For each client request, one of these files is selected according to the path selection probabilities, and delivered as if it were an ordinary linear media file. The corresponding tight lower bound on the required server bandwidth is given by

$$B_{min}^{path} = \sum_{i \in \mathcal{L}} \int_0^{T} \frac{\mathsf{dx}}{d + x + \frac{1}{\lambda_i}} = \sum_{i \in \mathcal{L}} \ln\left(\frac{p_i N}{p_i N \frac{d}{T} + 1} + 1 \right), \tag{4}$$

where $\mathcal{L}$ denotes the set of indices of the portions of the video file that are leaves in the tree structure, and where for notational convenience it is assumed that each complete path has the same playback time T.

The key observations from Fig. 2 are that: (1) multicast-based delivery techniques for non-linear media have the potential to yield large reductions in bandwidth requirements (note that with unicast, the required server bandwidth is N), and (2) techniques that exploit the particular non-linear structure, rather than treating each portion or path as a separate linear media file, have the greatest potential.

The potential bandwidth reductions from multicast-based delivery are dependent on the non-linear media structure. Fig. 3 shows the impact of increasing the height of a balanced binary tree structure, for fixed normalized request rate. As the height increases, the number of portions of the video file increases exponentially, as does the number of possible paths that clients may select from. Furthermore, relative to the total length of a path the length of each video portion decreases; i.e., branch points become more closely spaced. Not surprisingly, the potential benefits of multicast-based delivery decrease. (Similarly, these benefits also decrease when the branching factor is increased at each branch point, with fixed height, owing to the resulting increase in the number of paths.) However, even with a height of 10 and more than a thousand possible paths, multicast-based delivery still has the potential for an order-of-magnitude reduction in server bandwidth, assuming immediate service and the request rate considered in the figure. These potential bandwidth savings are explained largely by the potential for shared delivery of the video portions with the highest selection probabilities (i.e., those along popular paths or near the root).

C. Maximum Client Data Overhead

Without *a priori* knowledge that would rule out some path choices, achieving the lower bound of eq. 2 requires that a client listen to any multicast of data from a video portion that (at the time of the multicast) could still be on the client's eventual path. Since data is being multicast at minimum frequency, it is guaranteed that the same data is not multicast multiple times during the time that a client can obtain it. Thus, on average, the amount of data received from each video portion not on the client's eventual path is given by the rate at which data from that portion is multicast, times the length of the period over which the client can obtain such multicasts. The latter quantity for a client that follows the path to a leaf video portion i and for a video portion j that is not on this path (i.e., is not i or an ancestor of i), is equal to the sum of the start-up delay d and the playback durations of all video portions on the chosen path that are also on the path to j. This yields an average client data overhead, in units of the amount of video data on a complete path, of

$$\left(\sum_{i\in\mathcal{L}} p_i \sum_{j\in\overline{\mathcal{A}}(i)} \left(d + \sum_{k\in\mathcal{A}(i,j)} T_k\right) \ln\left(\frac{N_j}{N_j\frac{d+t_j}{T_j}+1}+1\right)\right) / \ T,$$

where $\overline{\mathcal{A}}(i)$ denotes the set of indices of those portions that are not portion i or an ancestor of portion i, and $\mathcal{A}(i,j)$ denotes the set of indices of those portions that are ancestors of both i and j.

Fig. 4 shows the average client data overhead incurred to achieve the lower bound of eq. 2 for balanced binary tree structures of various heights, immediate service, and no *a priori* knowledge of client path choices. Note that, for a given height tree, as the request rate increases the average client data overhead initially increases and then levels off since the lower bound server bandwidth for portion j has finite asymptote for all $j > 1$. Similarly, for fixed arrival rate, as the height increases the average client data overhead also increases. Finally, the data overhead when clients snoop on all portions that could still be on their eventual path can be significant, particularly when the tree height is greater than four and the normalized request rate is greater than 100.

IV. RESTRICTED SNOOP-AHEAD

Owing to client reception rate and/or buffer space limitations, the client data overheads shown in Fig. 4 may be infeasible. This section considers approaches in which clients snoop less aggressively on multicasts from video portions ahead of their current play point, thus reducing this overhead.

Snoop-ahead can be restricted in at least two basic ways. First, as considered in Section IV.A, restrictions may be based on distance from the current play point. Second as considered in Section IV.B, restrictions can be based on (a) overall path selection probabilities, or (b) client-specific path prediction, according to the past behavior of that client, client classification, and/or advance selection by the client.

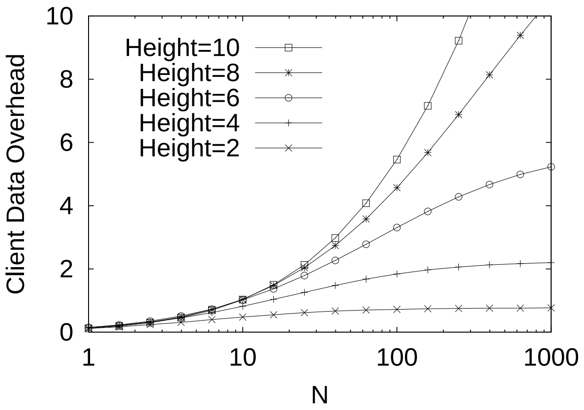


Fig. 4. Client Data Overhead for Unrestricted Snoop-ahead (α = 1, d = 0, no *a priori* knowledge of client path selection)

A. Distance-based Restricted Snoop-ahead

A simple approach that restricts snoop-ahead based on distance is to only snoop on multicasts from the current video portion (but ahead of the current play point), and from all portions following the next branch point.[1] Thus, with this approach, clients snoop on multicasts from each video portion i during playback of that portion, and, if not the initial, root portion (i.e., $i \geq 2$), during the playback of i's parent in the tree structure. A tight lower bound on the required server bandwidth for any technique utilizing this approach is given by

$$\begin{aligned} B^{next}_{min} &= \int_0^{T_1} \frac{\mathsf{dx}}{d + x + \frac{1}{\lambda}} + \sum_{i=2}^{V} \int_0^{T_i} \frac{\mathsf{dx}}{T_{a(i)} + x + \frac{1}{\lambda_i}} \\ &= \ln\left(\frac{N_1}{N_1\frac{d}{T_1}+1}+1\right) + \sum_{i=2}^{V} \ln\left(\frac{N_i}{N_i\frac{T_{a(i)}}{T_i}+1}+1\right), \end{aligned} \quad (5)$$

where $a(i)$ denotes the index of the immediate ancestor (parent) of i. Achieving this bound would incur an average client data overhead of

$$\left(\sum_{i=2}^{V} p_i T_{a(i)} \sum_{j\in\mathcal{S}(i)} \ln\left(\frac{N_j}{N_j\frac{T_{a(j)}}{T_j}+1}+1\right)\right) / \ T,$$

where $\mathcal{S}(i)$ denotes the set of indices of the siblings of i in the tree structure. Corresponding results can be derived for approaches in which clients snoop on transmissions from future video portions up to k branch points ahead, for some fixed $k > 1$.

[1]For clarity of presentation, we assume here and for the subsequent restricted snoop-ahead approaches, that prior to beginning playback, in the case of $d > 0$, clients only listen to multicasts from the initial, root portion of the video. The same analysis approach can be employed with alternative assumptions.

B. Client Path Prediction Approaches

With skewed branch selection probabilities, it may be possible to substantially reduce the client data overhead, with only a small cost in increased server bandwidth, by snooping on multicast transmissions from only the most popular portion of the video following the next branch point. The corresponding tight lower bound is given by

$$\begin{aligned} B_{min}^{popnext} &= \int_0^{T_1} \frac{\mathsf{d}x}{d+x+\frac{1}{\lambda}} + \sum_{i\in\mathcal{P}} \int_0^{T_i} \frac{\mathsf{d}x}{T_{a(i)}+x+\frac{1}{\lambda_i}} \\ &\quad + \sum_{i\in\overline{\mathcal{P}}} \int_0^{T_i} \frac{\mathsf{d}x}{x+\frac{1}{\lambda_i}} \\ &= \ln\left(\frac{N_1}{N_1\frac{d}{T_1}+1}+1\right) + \sum_{i\in\mathcal{P}} \ln\left(\frac{N_i}{N_i\frac{T_{a(i)}}{T_i}+1}+1\right) \\ &\quad + \sum_{i\in\overline{\mathcal{P}}} \ln\left(N_i+1\right), \qquad (6) \end{aligned}$$

where $\mathcal{P}$ and $\overline{\mathcal{P}}$ denote the set of indices of those portions of the video file that are the most popular, or are not the most popular, video portions among their siblings, respectively (excluding the root portion, which has no siblings). Achieving this bound would incur an average client data overhead of

$$\left(\sum_{i\in\overline{\mathcal{P}}} p_i T_{a(i)} \ln\left(\frac{N_{s(i)}}{N_{s(i)}\frac{T_{a(i)}}{T_{s(i)}}+1}+1\right)\right) \Big/ \ T,$$

where $s(i)$ denotes the index of the most popular sibling of video portion i.

Rather than just snooping on transmissions from the most popular video portion after the next branch point, clients could snoop on transmissions from all video portions on the most popular path from the current position to a leaf. The corresponding tight lower bound is given by

$$\begin{aligned} B_{min}^{poppath} &= \int_0^{T_1} \frac{\mathsf{d}x}{d+x+\frac{1}{\lambda}} + \sum_{i\in\mathcal{P}} \int_0^{T_i} \frac{\mathsf{d}x}{\sum\limits_{j\in\mathcal{U}(i)} T_j+x+\frac{1}{\lambda_i}} \\ &\quad + \sum_{i\in\overline{\mathcal{P}}} \int_0^{T_i} \frac{\mathsf{d}x}{x+\frac{1}{\lambda_i}} \\ &= \ln\left(\frac{N_1}{N_1\frac{d}{T_1}+1}+1\right) + \sum_{i\in\mathcal{P}} \ln\left(\frac{N_i}{N_i\frac{\sum_{j\in\mathcal{U}(i)} T_j}{T_i}+1}+1\right) \\ &\quad + \sum_{i\in\overline{\mathcal{P}}} \ln\left(N_i+1\right), \qquad (7) \end{aligned}$$

where $\mathcal{U}(i)$ denotes the set of indices of ancestors on the path back towards the root from i (not including i itself), up to and including the first portion that is not the most popular among its siblings. (If there is no such portion on this path, the set includes the indices of all ancestors on the path back to and including the root.) Achieving this bound would incur an average client data overhead of

$$\left(\sum_{i\in\overline{\mathcal{P}}} p_i \Big(\sum_{j\in\mathcal{U}(i)} T_j\Big) \sum_{j\in\mathcal{D}(a(i))} \ln\left(\frac{N_j}{N_j\frac{\sum_{k\in\mathcal{U}(j)} T_k}{T_j}+1}+1\right)\right) \Big/ \ T,$$

where $\mathcal{D}(a(i))$ denotes the set of indices of video portions on the most popular path down to a leaf from (but not including) the parent of portion i.

Consider now the case in which more accurate client-specific path prediction is possible, and clients snoop on multicasts from all video portions on their predicted (rather than the overall most popular) path from the current position to a leaf. Analysis of this approach requires a model of path prediction accuracy. Here we use a very simple model in which branch choices with selection frequency (conditional on reaching the respective branch point) at least equal to a parameter f are always successfully predicted, and less popular branch choices are never predicted. The corresponding tight lower bound is given by

$$\begin{aligned} B_{min}^{pred} &= \int_0^{T_1} \frac{\mathsf{d}x}{d+x+\frac{1}{\lambda}} + \sum_{i\in\mathcal{F}} \int_0^{T_i} \frac{\mathsf{d}x}{\sum\limits_{j\in\mathcal{W}(i)} T_j+x+\frac{1}{\lambda_i}} \\ &\quad + \sum_{i\in\overline{\mathcal{F}}} \int_0^{T_i} \frac{\mathsf{d}x}{x+\frac{1}{\lambda_i}} \\ &= \ln\left(\frac{N_1}{N_1\frac{d}{T_1}+1}+1\right) + \sum_{i\in\mathcal{F}} \ln\left(\frac{N_i}{N_i\frac{\sum_{j\in\mathcal{W}(i)} T_j}{T_i}+1}+1\right) \\ &\quad + \sum_{i\in\overline{\mathcal{F}}} \ln\left(N_i+1\right), \qquad (8) \end{aligned}$$

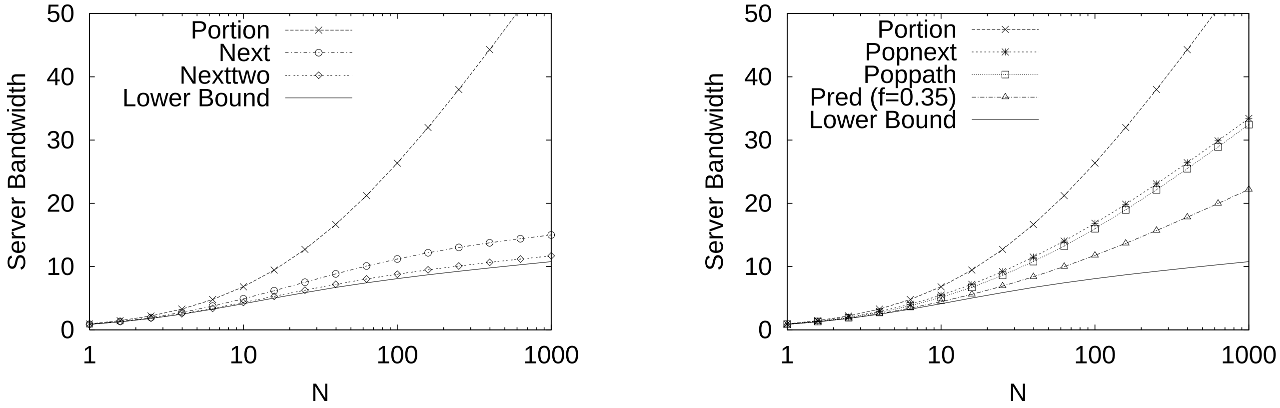
where $\mathcal{F}$ and $\overline{\mathcal{F}}$ denote the set of indicies of those portions of the video file whose conditional selection frequency is at least f, or less than f, respectively, and $\mathcal{W}(i)$ denotes the set of indicies of ancestors on the path back towards the root from i (not including i itself), up to and including the first portion that is a member of the set $\overline{\mathcal{F}}$. (If there is no such portion on this path, the set includes the indices of all ancestors on the path back to and including the root.) Achieving this bound would incur an average client data overhead of

$$\frac{\sum\limits_{i\in\overline{\mathcal{F}}} p_i \Big(\sum\limits_{j\in\mathcal{W}(i)} T_j\Big) \sum\limits_{l\in\mathcal{L}(\mathcal{S}(i))} \frac{p_l}{\sum_{m\in\mathcal{L}(\mathcal{S}(i))} p_m} \sum\limits_{j\in\mathcal{D}(a(i),l)} B_{j\ min}^{pred}}{T},$$

where $\mathcal{L}(\mathcal{S}(i))$ denotes the set of indices of leaf video portions in the collection of subtrees rooted at siblings of i for which the path from that sibling includes only video portions in the set $\mathcal{F}$, $\mathcal{D}(a(i),l)$ denotes the set of indices of video portions on the path down to leaf portion l beginning from (but not including) the parent of i, $B_{j\ min}^{pred}$ denotes the bandwidth used for multicasts of video portion j, as given by the term for video portion j on the right-hand side of eq. 8, and where we have assumed that an incorrect path prediction is for each of the paths that could have been predicted with probability proportional to its relative popularity.

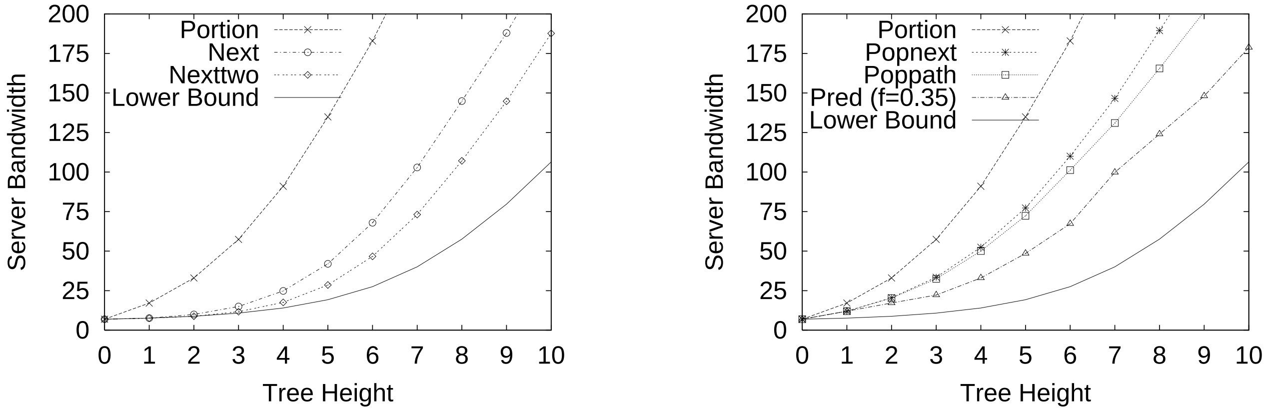
C. Policy Comparisons

Figs. 5 and 6 graph the bandwidth expressions given above as functions of the request rate (for the same binary tree structure assumed for Fig. 2), and the tree height (for fixed request rate), respectively. Also shown is the server bandwidth for the approach (*nexttwo*) in which clients snoop on multicasts



(a) Distance-based Policies (b) Prediction-based Policies

Fig. 5. Performance with Restricted Snoop-ahead (balanced binary tree with height 3, α = 1, d = 0)



(a) Distance-based Policies (b) Prediction-based Policies

Fig. 6. Impact of Tree Height on Restricted Snoop-ahead Performance (α = 1, N = 1000, d = 0)

from the current video portion plus from all portions following the next and next two branch points, which is derived similarly to eq. 5. For comparison purposes, the figures also show the server bandwidth for unrestricted snooping (i.e., the lower bound of eq. 2), and for the approach in which each portion is treated as a separate linear video file (*portion*). Corresponding results for the client overhead are given in Figs. 7 and 8.

Consider first the results for *portion*, *next*, *nexttwo*, and unrestricted snooping. With *portion*, clients only listen to multicasts of data from the video portion currently being played. Snooping of multicasts of data from beyond the next branch point (*next*) yields a large reduction in server bandwidth. Snooping farther ahead, as in *nexttwo*, yields diminishing returns. As seen by the results in Fig. 6(a), for trees of low to moderate height *nexttwo* has minimal required server bandwidth fairly close to the lower bound of eq. 2. The results in Figs. 5(a), 6(a), 7, and 8 show that the *next* and *nexttwo* approaches can often achieve large reductions in average client data overhead compared to the unrestricted snooping approach, at fairly modest cost in server bandwidth.

The *popnext*, *poppath*, and *pred (f=0.35)* approaches use *a priori* information regarding client path selection in an attempt to achieve a better tradeoff between server bandwidth and client overhead. Although *popnext* and *poppath* achieve low client overhead, as seen in Figs. 7 and 8, they achieve poorer server bandwidth scalability than *next* and *nexttwo*. These results show that very approximate client path prediction, such as occurs with *popnext* and *poppath* at branch points at which the branch selection probabilities are not highly skewed, is not as effective in reducing server bandwidth as is snooping on all multicasts of data that could be needed soon, as in *next*. In contrast, the more accurate *pred (f=0.35)* approach achieves lower client data overhead than *next* and comparable server bandwidth scaling. Finally, note that the approaches in which clients snoop on multicasts from all video portions on a path from the current position to a leaf (*poppath* and *pred*) become relatively more attractive with respect to their server bandwidth usage, and relatively less attractive with respect to client data overhead, for high tree heights.

Hybrid approaches may perform even somewhat better under some conditions. For example, consider a branch point at which one choice is highly popular and the other is much less popular. Clients could snoop on multicasts from both of these video portions (as in *next*), while also predicting a

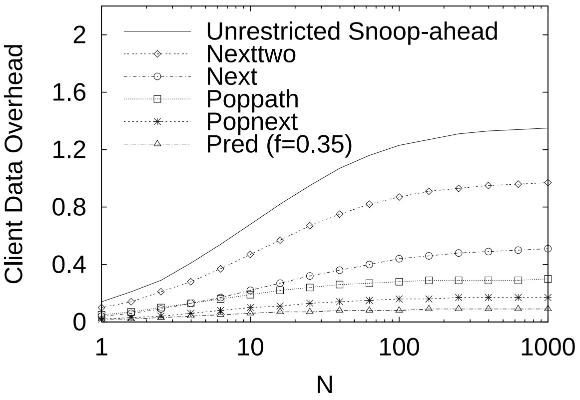


Fig. 7. Client Overhead with Restricted Snoop-ahead (balanced binary tree with height 3, α = 1, d = 0)

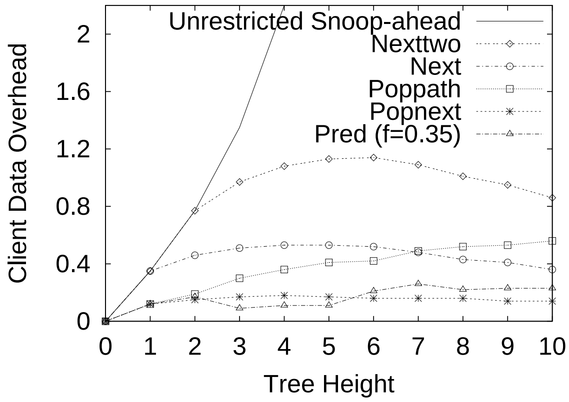
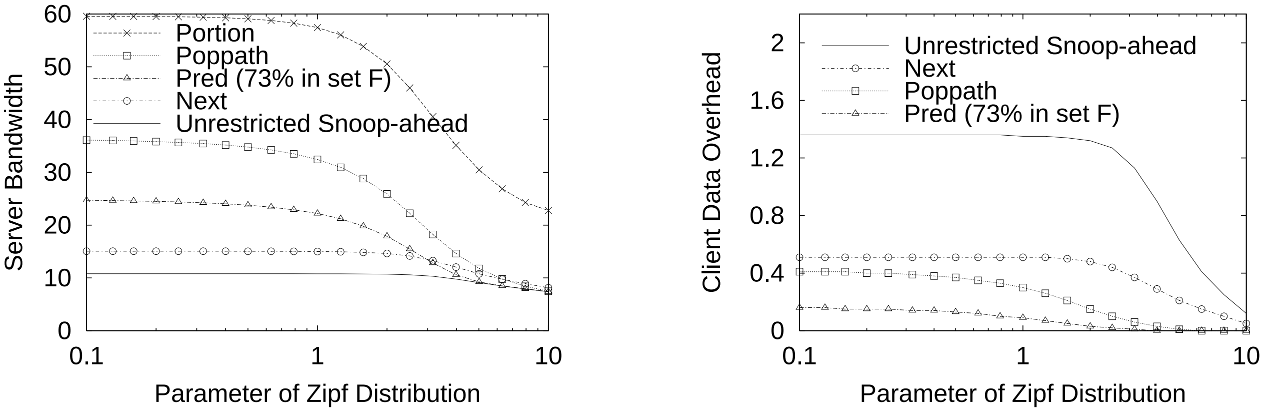


Fig. 8. Impact of Tree Height on Overhead (α = 1, N = 1000, d = 0)



(a) Server Bandwidth

(b) Client Overhead

Fig. 9. Sensitivity to Skewness in Selection Probabilities (balanced binary tree with height 3, N = 1000, d = 0)

path that includes the highly popular choice and snooping on multicasts from subsequent video portions (as in *poppath* or *pred*). Preliminary investigations of a hybrid of the *next* and *pred* approaches confirm this intuition.

Fig. 9 shows the sensitivity of the above results to skewness in the leaf selection probabilities, specifically to the Zipf parameter α. (Curves for *nexttwo* and *popnext* have been omitted but have similar form.) For *pred*, the parameter f has been varied so that the percentage of video portions in the set $\mathcal{F}$ is constant, equal to that with $f = 0.35$ and $\alpha = 1$. Thus, for *pred* (as well as for *poppath*), the number of relatively popular video portions whose selection is successfully predicted remains constant as α varies. Note that there is relatively little variation in the required server bandwidth and client data overhead for each approach for $\alpha \leq 1$ (i.e., for no skew to moderately high skew). As α increases beyond one, the server bandwidth and client data overhead for each approach decrease substantially. A key conclusion is that the simple *next* approach, and the *pred* approach with correct path predictions for at least 75% of the video portions, achieve an attractive trade-off between required server bandwidth and client data overhead, over a wide range of α values.

V. Scalable Delivery Protocols

A. Hierarchical Stream Merging

Hierarchical stream merging (HSM) protocols [5], as applied to linear media, start a new transmission of the media file for each client request. In the simplest type of HSM, each client also listens to the closest active earlier stream, so that its own stream can terminate after transmitting the data that was missed in the earlier stream. At that point, the clients associated with the two streams are said to be "merged" into a single "group", which can then go on to merge with other groups.

Extending HSM to non-linear media requires a more dynamic notion of client group, since clients that merge while listening to one video portion may take different paths at the next branch point, thus splitting the group. Also, a more complex policy may be required for determining what stream a client listens to, in the case where the closest earlier stream is beyond the next branch point. It would seem that, in this case, the client should listen to the closest earlier stream currently delivering data from the path that the client will select, should such a stream exist and should the branch choice be known or accurately predicted.

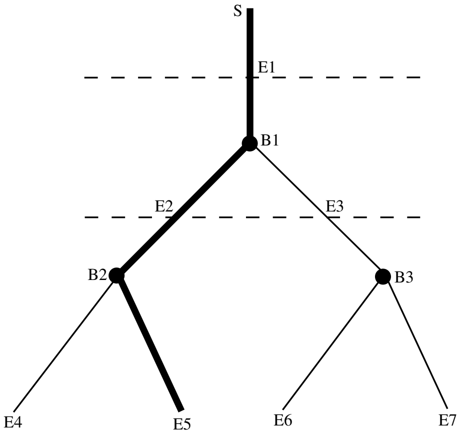


Fig. 10. OPB-KP Segment Partitioning for an Example Media Structure ("S" label is for start of media, "Bi" labels are for branch points, "Ei" labels are for segment end points, dashed lines indicate segment boundaries, K = 3, s = 2, r = 1)

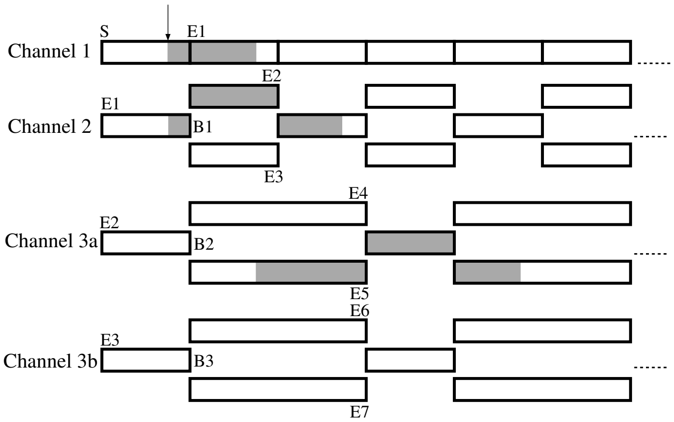


Fig. 11. OPB-KP Channels for Structure of Fig. 10 (shaded areas are listening periods of example client, K = 3, s = 2, r = 1)

The results from Section IV suggest that using overall path selection probabilities to guide which earlier stream a client listens to when the closest earlier stream has past the next branch point may not be the best strategy. This intuition is confirmed by simulation results showing that listening to the closest earlier stream (on or past the same video portion, regardless of which branch it may be on if beyond a branch point) yields slightly better performance than listening to the closest stream on the most popular branch [17]. Section V.C presents simulation results for both this *HSM-Unknown Path* (HSM-UP) protocol in which clients listen to the closest earlier stream, and for *HSM-Known Path* (HSM-KP) in which it is assumed that precise client-specific path prediction is possible, and thus clients can listen to the closest earlier stream delivering data from the path they will select. Note that with HSM-KP, clients belonging to the same "group" may be listening to different earlier streams. With both protocols, a group may split as the clients within a group reach a branch point, in which case the server will need to start additional stream(s) so that there is one stream per path followed. These characteristics also complicate merging behavior. In the simulations from which results are presented here, it is assumed that when a client or group of clients merges with an earlier group, all clients in the earlier group restart listening to earlier stream(s), as in HSM for linear media. Other options are investigated in [17].

B. Optimized Periodic Broadcast

The periodic broadcast protocols that we develop here for non-linear media are based on the optimized periodic broadcast (OPB) protocols described in [12]. In these protocols, as applied to linear media, the media file is partitioned into K segments, with each segment being repeatedly multicast on a separate channel at rate r. Clients are assumed able to simultaneously listen to a maximum of s channels. The segment size progression is computed such that each segment is received just in time for playback if clients begin listening to the s channels delivering the first s segments immediately, begin listening to the channel for segment k ($k > s$) immediately after fully receiving segment $k - s$, and begin playback after reception of the first segment is complete.

For the case in which client path selection is known *a priori*, we propose a variant of OPB called *OBP-Known Path* (OPB-KP). Each complete path through the non-linear media file is partitioned using the same segment size progression as in OBP for linear files. Shared portions of paths share the corresponding segments. (We assume here that if the file has a directed acyclic graph structure, then the path lengths to any video portion with multiple parents are identical.) If a segment crosses a branch point, the data from each media portion after the branch point is delivered on a separate sub-channel, each at rate r. Thus, for such a segment, the server will repeatedly first transmit the data from before the branch point (at rate r), and then transmit the data from after the branch point (at total rate r times the number of portions after the branch point). Each client listens to the channels and sub-channels appropriate to its path. Fig. 11 shows the channels used in the OPB-KP protocol for the example non-linear video structure shown in Fig. 10, assuming each path is partitioned into three segments (K = 3), clients listen to two channels concurrently (s = 2), and segments are transmitted at the playback data rate (r = 1). Also shown are the periods during which an example client listens to the transmissions on each channel, assuming the client request arrives at the point indicated and that the client takes the path shown in Fig. 10. As shown by the results in Section V.C, if it is assumed that the server can detect if there are any listeners on a channel (or sub-channel), and stop transmitting on the channel if not, this scheme is efficient.

For the case in which client path selection decisions are known only when they are made at the respective branch points, our key insight is that periodic broadcast is still feasible as long as any segment that a client begins to download prior to a branch point, and that includes data from after the branch point, includes the respective data from all of the branches.

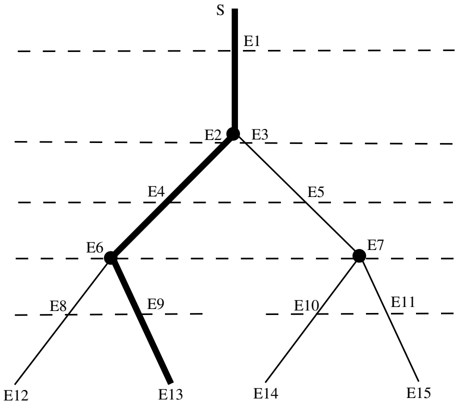


Fig. 12. OPB-UP Segment Partitioning for an Example Media Structure ("S" label is for start of media, "Ei" labels are for segment end points, dashed lines indicate segment boundaries, K = 6, s = 2, r = 1)

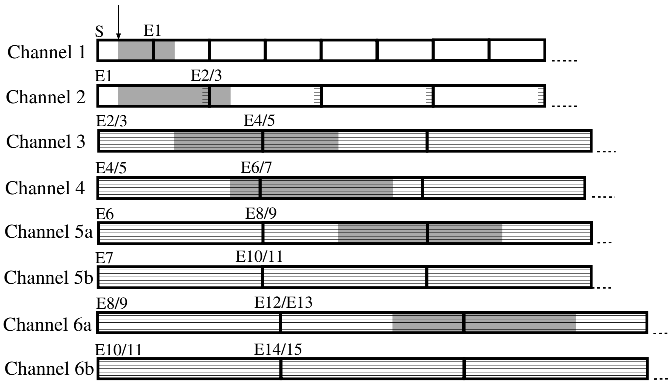


Fig. 13. OPB-UP Channels for Structure of Fig. 12 (shaded areas are listening periods of example client, striped areas indicate multiplexed transmissions, K = 6, s = 2, r = 1)

Specifically, suppose that between when a client begins to listen to the transmission of a particular segment k and the beginning of playback of that segment, video playback does not cross a branch point. If segment k itself also does not cross a branch point, then it must be part of the same video portion that was being played back during its reception. If, on the other hand, segment k does cross a branch point, then it must include some of the video portion prior to the branch point (as determined by the segment starting position), plus a fraction of *each* video portion after the branch point (as determined by the segment ending position). Note that the playback duration of such a segment will be less than what its size in bytes (and corresponding transmission time) would suggest, since the client will playback only the data on its chosen path. Suppose now that between when a client begins to listen to the transmission of a segment and the beginning of playback of that segment, video playback does cross a branch point. In this case, the entire segment multiplexes data from multiple paths, as the segment begins after the branch point and it is unknown which branch a client will take.

Fig. 13 shows the channels used in this *OBP-Unknown Path* (OPB-UP) protocol for the example non-linear video structure shown in Fig. 12, assuming each path is partitioned into six segments (K = 6), clients listen to two channels concurrently (s = 2), and segments are transmitted at the playback data rate (r = 1). Also shown are the periods during which an example client listens to the transmissions on each channel, assuming the client request arrives at the point indicated in the figure and that the client takes the path shown in Fig. 12.

Feasible segment sizes for OPB-UP can be computed using the algorithm outlined in Fig. 14. Although this algorithm is designed for balanced binary trees, it can be extended for more general types of media structures. Here l_k denotes the playback duration of segment k, u_k denotes the time when a client begins reception of the segment, measured relative to the start of the video file playback, e_k denotes the latest time by which a client can end reception of the segment, measured relative to the start of video playback (also equal to the playback point corresponding to the beginning of the segment), y_k denotes the segment transmission time when the segment is of maximal length, and w_k denotes the playback point corresponding to the end of the segment in the case in which the segment does not encounter a branch point. The outer loop attempts to find the start-up delay (transmission time of the first segment) such that the cumulative length of K segments (where K is given as an input) matches the length of a complete path. The algorithm makes the simplifying restriction that no segment can have a multiplexing level of more than two (i.e., include data from more than two paths), and the assumption that the first segment does not cross any branch points. It further assumes that branch points are never sufficiently close together that a zero length is computed for a segment (as would occur in case 2.2 when the branch point B is at e_k), although it could be extended to handle this case by simply delaying beginning reception of the segment until after the next branch choice has been made. Such delays could be more generally beneficial, as well, but the algorithm in Fig. 14 simply assumes that a client begins reception of a new segment (if any remain) immediately after reception of a previous segment completes. The design of optimal periodic broadcast protocols for various types of non-linear media structures is left for future work.

C. Performance Comparisons

Figs. 15, 16, and 17 show the server bandwidth used by the HSM and OPB protocols for non-linear media streaming, together with the analytic lower bound from eq. 2. HSM results are from simulation. The results for the OPB variants are obtained under the assumption that transmission on a channel is stopped whenever no client is listening to that channel. (The probability that no client is listening to a channel can be easily computed under the assumption of Poisson request arrivals, which are also assumed in the simulation of HSM.)

For HSM-KP and OPB-KP, path prediction is assumed to be perfect. For HSM, imperfect path prediction would yield results intermediate between the results for HSM-UP and HSM-KP. For OPB-KP, an error in path prediction would be more difficult to recover from. All data is received prior to

Function $Partition(K, s, r)$
1. For ($i = 0$; $i \leq M$; i++)
2. $d = \epsilon_1 + i(T_1/r - \epsilon_1)/M$
3. $Compute\_Segsize(K, d, s, r)$
4. If ($\left|\sum_{j=1}^{K} l_k - T\right| \leq \epsilon_2$)
5. Return $Success$
6. End For
7. Return $Failed$
End Function

Procedure $Compute\_Segsize(K, d, s, r)$
8. $l_1 = dr$
9. For ($k = 2$; $k \leq K$; k++)
10. $u_k = -d, \ \ k \leq s$;
$u_k = s$'th latest of $\{ \ u_j + l_j/r \mid 1 \leq j < k \ \}, \quad k \geq s+1$
11. $e_k = \sum_{j=1}^{k-1} l_j$; $y_k = e_k - u_k$
12. Case 1: no branch point in (u_k, e_k)
13. $w_k = y_k r + e_k$
14. Case 1.1: no branch point in $[e_k, w_k)$
15. $l_k = y_k r$
16. Case 1.2: first branch point in $[e_k, w_k)$ is at B and no branch point in $(B, B + (y_k r - (B - e_k))/2)$
17. transmit interleaved data after branch point
18. $l_k = B - e_k + (y_k r - (B - e_k))/2$
19. Case 1.3: first branch point in $[e_k, w_k)$ is at B, and first branch point in $(B, B + (y_k r - (B - e_k))/2)$ is at B_2
20. segment ends at branch point B_2
21. $l_k = B_2 - e_k$
22. Case 2: one branch point in (u_k, e_k)
23. transmit interleaved data
24. $w_k = y_k r/2 + e_k$
25. Case 2.1: no branch point in $[e_k, w_k)$
26. $l_k = y_k r/2$
27. Case 2.2: first branch point in $[e_k, w_k)$ is at B
28. segment ends at branch point
29. $l_k = B - e_k$
30. End For
End Procedure

Fig. 14. Algorithm for OPB-UP Segment Sizes (balanced binary tree)

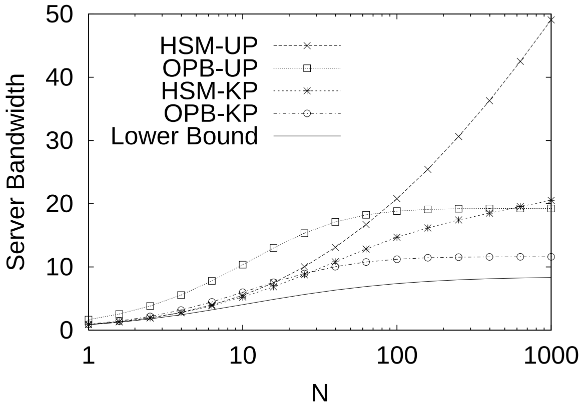


Fig. 15. Performance of Scalable Delivery Protocols (balanced binary tree with height 3, α = 1, d = 0.01 for OPB and lower bound, r = 0.25, s = 8)

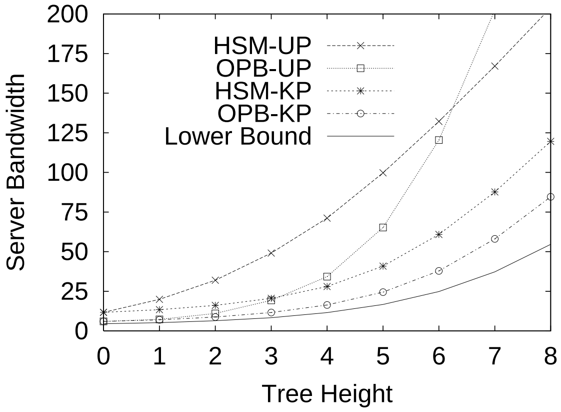


Fig. 16. Performance with Varying Height (α = 1, N = 1000, d = 0.01 for OPB and lower bound, r = 0.25, s = 8)

its playback time. A client whose path is mispredicted will have listened to transmissions of the wrong data from after the mispredicted branch point. Recovery would require either interruption in playback (so as to allow time for the client to receive the data that it would have received by this point, had the branch choice been correctly predicted), or use of a unicast stream that would deliver data sequentially from the branch point at rate at least equal to the playback data rate.

The key observations from these figures are: (1) stopping transmission on a channel when there are no clients listening allows periodic broadcast performance to be competitive even under light load, (2) precise path prediction yields a large improvement in performance, (3) OPB-KP yields performance essentially as close to the lower bound from eq. 2 as could be expected, given that the former assumes that each client can only receive data at each point in time at a total aggregate rate of twice the streaming rate ($r \times s = 2$), whereas the latter places no such restriction (see [5], [12] regarding the impact of client receive rate limitations), and (4) the precise relative performance of the HSM and OPB variants depends on request arrival rate and the client start-up delay used in OPB (note that HSM provides immediate service, although variants that use a batching start-up delay have also been proposed [4]). The results for HSM-UP and HSM-KP suggest that it may be fruitful to investigate HSM variants in which clients may snoop on multiple earlier streams (e.g., one for each possible choice at the next branch point, similar to *next*), or in which client-specific path prediction is employed (as in *pred*).

D. Prototype Implementation

A rudimentary implementation of scalable non-linear media streaming has been added to the SWORD prototype streaming system [12]. The SWORD system consists of server and client components, built using the open source Apache proxy server code as a base. These interpose between Windows Media players and servers, and replace the normal unicast delivery with multicast delivery using hierarchical stream merging.

Our implementation of non-linear media streaming stores

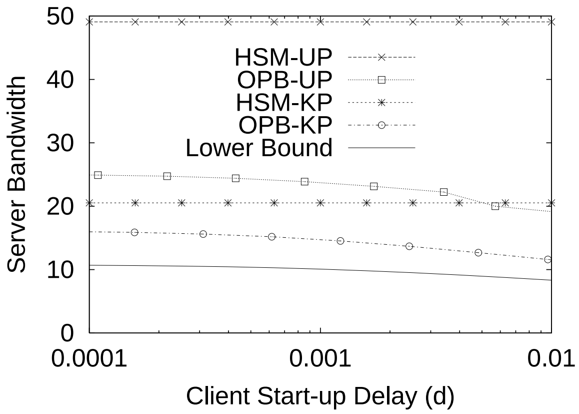


Fig. 17. Impact of OPB Start-up Delay
(balanced binary tree with height 3, α = 1, N = 1000, r = 0.25, s = 8)

each portion of the non-linear structure as a separate file. Modification of header fields and spoofing of requests by the SWORD client component allow this structure to be invisible to the client player, to which it appears that only a single video file is being played (transitions between the video portions are seamless). Dynamic client path selection is currently supported through a web page interface. The present implementation uses built-in knowledge of the media file structure; on-going work concerns description of non-linear media structures in meta files. Our implementation has demonstrated that non-linear media streaming can be implemented relatively easily, even in the context of commercial media streaming systems.

VI. Conclusions

This paper has considered "non-linear" video content in which clients can tailor their video stream according to individual preferences, within the constraints of a predefined tree or graph structure. Tight lower bounds on server bandwidth were developed that show the potential for bandwidth reduction using multicast delivery in the context of non-linear media, as well as illuminate the advantages/disadvantages of various approaches to client snoop-ahead and the benefits of *a priori* path knowledge. The key insights from the bounds analysis are (1) correct client path predictions for more than 75% of the video portions greatly reduces the required server bandwidth with very modest client data overhead, and (2) in the absence of fairly precise *a priori* information about client path selections, a simple policy in which clients only listen to transmissions from their current video portion and those immediately following the next branch point, achieves better server bandwidth scalability than using overall path selection probabilities to determine which transmissions to listen to.

New stream merging and periodic broadcast protocols were devised, in part using insight from our bounds analysis, and shown to achieve much of the potential bandwidth savings. The new periodic broadcast protocols were found to be competitive with the new stream merging protocols at all request rates, assuming that in the former protocols the server transmits on a channel only when at least one client is listening. On-going research is focussed on further analysis of hybrid protocol classes, improved stream merging and periodic broadcast protocols, and further development and experimentation with a prototype delivery system.

Acknowledgements

We thank David Sundaram-Stukel and Jeremy Parker for their work on the SWORD prototype, Peter O'Donovan for his work on implementing seamless playback of non-linear media, Brian Gallaway for his assistance with the prototype software and hardware, and Wenguang Wang and the anonymous referees for their constructive feedback on the paper.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proc. IEEE ICMCS'96*, pages 118–126, Hiroshima, Japan, June 1996.

[2] Y. Birk and R. Mondri. Tailored transmissions for efficient near-video-on-demand service. In *Proc. IEEE ICMCS'99*, pages 226–231, Florence, Italy, June 1999.

[3] S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. In *Proc. IEEE ICCCN'97*, pages 200–207, Las Vegas, NV, Sept. 1997.

[4] D. L. Eager, M. K. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective video-on-demand. In *Proc. IS&T/SPIE MMCN'00*, pages 206–215, San Jose, CA, Jan. 2000.

[5] D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, Sept./Oct. 2001.

[6] L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proc. NOSSDAV'98*, pages 317–329, Cambridge, UK, July 1998.

[7] L. Gao and D. Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proc. ICMCS'99*, pages 117–121, Florence, Italy, June 1999.

[8] A. Hu. Video-on-demand broadcasting protocols: A comprehensive study. In *Proc. IEEE INFOCOM'01*, pages 508–517, Anchorage, AL, Apr. 2001.

[9] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *Proc. ACM MULTIMEDIA'98*, pages 191–200, Bristol, U.K., Sept. 1998.

[10] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proc. ACM SIGCOMM'97*, pages 89–100, Cannes, France, Sept. 1997.

[11] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.

[12] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. *IEEE/ACM Trans. on Networking*, 11(2):195–209, Apr. 2003.

[13] I. Nikolaidis, F. Li, and A. Hu. An inherently loss-less and bandwidth-efficient periodic broadcast scheme for VBR video. In *Proc. ACM SIGMETRICS'00*, pages 116–117, Santa Clara, CA, June 2000.

[14] J.-F. Pâris. A broadcasting protocol for compressed video. In *Proc. EUROMEDIA'99*, pages 78–84, Munich, Germany, Apr. 1999.

[15] S. Sen, L. Gao, and D. Towsley. Frame-based periodic broadcast and fundamental resource tradeoffs. Technical Report 99-78, Comp. Sci. Dept., Univ. of Massachusetts Amherst, 1999.

[16] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *ACM Multimedia Systems J.*, 4(3):197–208, Aug. 1996.

[17] Y. Zhao. *Scalable Streaming of Stored Complex Multimedia*. Ph.D. Dissertation, Univ. of Saskatchewan, 2004.

[18] Y. Zhao, D. L. Eager, and M. K. Vernon. Efficient delivery techniques for variable bit rate multimedia. In *Proc. IS&T/SPIE MMCN'02*, pages 142–155, San Jose, CA, Jan. 2002.

[19] Y. Zhao, D. L. Eager, and M. K. Vernon. Network bandwidth requirements for scalable on-demand streaming. In *Proc. IEEE INFOCOM'02*, pages 1119–1128, New York, NY, June 2002.