Scalable On-Demand Streaming of Non-Linear Media

Yanping Zhao, Derek L. Eager, and Mary K. Vernon

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. For on-demand streaming of non-linear media, this paper analyzes the minimal server bandwidth requirements, and proposes and evaluates practical scalable delivery protocols.

1 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 clients to dynamically select among alternative parallel sequences of frames during playback, may enable new streaming media applications as well as enrich existing applications. 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, 8, 10] and hierarchical stream merging [4–6], as well as periodic broadcast protocols [1, 7, 9, 11–13, 17]. 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 receive and buffer data ahead of when it is needed for playback. A common mechanism is for clients to listen to one or more earlier streams in addition to their own stream, enabling them to "catch up" to the earlier clients. In periodic broadcast protocols, the video file is segmented, and each segment is repeatedly broadcast/multicast on one of a number of channels (or IP multicast addresses) according to some protocol-dependent transmission schedule. As in the immediate service protocols, clients receive and buffer data ahead of when it is needed for playback. Unlike the immediate service protocols, periodic broadcast protocols require clients to wait before beginning playback. This start-up delay is determined by the duration of the initial segment transmission. For "whole file" 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 explores scalable multicast streaming techniques for on-demand delivery of non-linear stored video. We first examine a basic question, namely to what extent does the potential benefit of multicast delivery diminish as the diversity in the data each client receives increases. This question is addressed by developing tight lower bounds on the server bandwidth required (for any protocol) as a function of file request rate and client start-up delay, for non-linear media files with varying path diversity. The results indicate that the potential bandwidth savings can be substantial, even for videos with high path diversity.

For non-linear videos, receiving data ahead of when it is needed, as is required in scalable streaming protocols, is complicated by uncertainty regarding which branch a client will follow at each branch point. There is a key 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 this tradeoff by deriving tight lower bounds on the server bandwidth required for various classes of protocols. Each protocol class considered makes use of a specific type of (partial) information about which branch a client will fol-

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.

low at each branch point. We consider the use of measured (over all clients) branch choice frequencies, as well as client-specific information as might result from pre-declaration of intended client paths or from client classification.

The server bandwidth bounds for each protocol class show that fairly precise *a priori* information regarding client path selection can dramatically reduce server bandwidth requirements as well as the overhead of delivering and buffering data that is never used. In the absence of such information, protocols that restrict how much data clients receive in advance of knowing whether or not it will be needed, based solely on how far ahead that data is in the video file, can greatly reduce the client data overhead at relatively small server bandwidth cost.

Using the 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. Variants of each type of protocol are developed that make differing assumptions concerning the availability of *a priori* path selection information. As with the lower bounds, precise *a priori* information regarding client path selection can substantially reduce the server bandwidth requirements for practical scalable protocols.

We assume constant bit rate video. Generalizations for variable bit rate video can be developed using similar approaches as for linear media [14, 15, 19].

The remainder of the paper is organized as follows. Section 2 describes models of non-linear media. A tight lower bound on the server bandwidth required for any protocol as a function of file request rate and client start-up delay is derived in Section 3. Achieving this lower bound without *a priori* client path selection information requires that each client listen to any multicast of data that it has not already received, and that is from a video portion reachable from the client's current play point. Also derived in Section 3 is the minimum client data overhead with this approach. Section 4 derives the server bandwidth bounds and associated client data overheads for various approaches that restrict the data that clients receive ahead of when it is needed. Section 5 presents new stream merging and periodic broadcast protocols for non-linear media, and comparative performance results. Section 6 concludes the paper.

2 Non-Linear Media Models

2.1 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 alternative possible ending portions. In a "complete path" playback of the video, the client plays the common portion followed by one of the ending portions. If the desired variant of the ending portion is chosen sufficiently ahead of the branch point (i.e., the end of the common initial portion), the complete path can be played without jitter. Unless otherwise stated, we assume 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 consists of the common root portion, plus all other portions on a path from the root 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 even a general graph structure.

The bounds in Sections 3 and 4 are developed for tree structures and assume that each client request is for a complete path playback, although the analysis can be generalized. The new immediate service protocols developed in Section 5.1 are applicable to non-linear media having a general graph structure, while the new periodic broadcast protocols in Section 5.2 are applicable to general tree structures and to directed acyclic graphs in which the path lengths to any video portion with multiple parents are identical. For clarity in the policy comparisons, 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.

2.2 Path Popularities

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 and complete path playback, we have explored several alternative popularity models. The model used for most of the numerical results presented in the paper assigns Zipf-distributed selection probabilities to leaves, 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.

Two other models that were evaluated include: (1) 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 (2) 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 $\frac{1}{1+1/2^\alpha}$, and the other with probability $\frac{1/2^\alpha}{1+1/2^\alpha}$ where α is the parameter of the Zipf distribution). These two models differ from the first model in that they have more skewed selection probabilities at the branch points near the root of the tree and less skewed probabilities at branch points near the leaves. However, as illustrated for one of these other models in Section 4.3, all three models were found to yield very similar results.

2.3 An Example

Fig. 1 shows a sample non-linear video structure. Each portion of the video is denoted by a node in the tree, and each branch is labelled with its selection probability. The structure in Fig. 1 corresponds to a balanced binary tree of height 3. Branch probabilities in the figure were computed by choosing Zipf-distributed leaf selection probabilities with the popularity ordering randomly determined, and then working up the tree. Also shown 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.

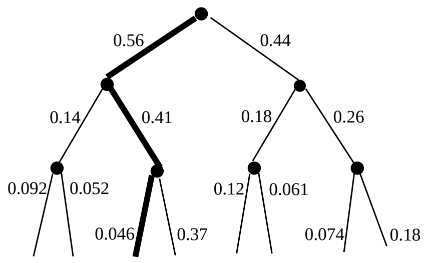


Figure 1: Example of a Non-Linear Media Structure

2.4 Path Prediction

Three scenarios are of interest with respect to the system's ability to predict a client's path selection: (1) no *a priori* knowledge is available of the likely path through the video that a particular client will take; (2) only the 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 scenario, 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. For the third scenario, we consider in Section 4.2 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 branches that could have been (incorrectly) predicted with probability proportional to its relative popularity.

3 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. Note that we consider only techniques that achieve server bandwidth savings without reducing the video quality. That is, in the absence of packet loss, each client receives exactly the same video data for each video portion that it receives as in a unicast system, and thus quality measures such as PSNR are unaffected. Packet loss recovery can be achieved using techniques such as those proposed in previous work for multicast-based on-demand delivery of linear media [13].

All results presented in this section and in Section 4 are analytic, based on the bounds developed in these sections. Section 3.1 defines the server bandwidth and data overhead performance metrics and outlines the analysis approach. In Section 3.2, a tight lower bound on the server bandwidth requirement is derived. Section 3.3 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 reduce the client data overhead are considered in Section 4. The notation used is defined in Table 1.

3.1 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 [20]. Also of interest is the average client data overhead, which is 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.

As noted previously, we assume constant bit rate video. Our analysis can be extended to variable bit rate video by modelling such video using concatenations of constant bit rate sections, as in previous work for linear media [19]. In our assumed context of constant bit rate video, we can express the required server bandwidth in units of the playback data rate. (In the context of variable bit rate video, the required server bandwidth can be expressed in units of the average playback data rate.)

Our lower bound analysis follows the same basic approach as has been used previously for linear media [2,6,7,16]. 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

Table 1: 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

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

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

This bound can be generalized to a broad class of non-Poisson arrival processes, yielding a similar result with difference bounded by a constant [6]. Bounds for non-linear media are derived below by applying similar analyses.

3.2 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

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

The solid curve with no symbols in 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 with $\alpha = 1$. The figure shows results for one particular randomly determined popularity ordering of the leaves. Alternative popularity orderings 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*), there is no

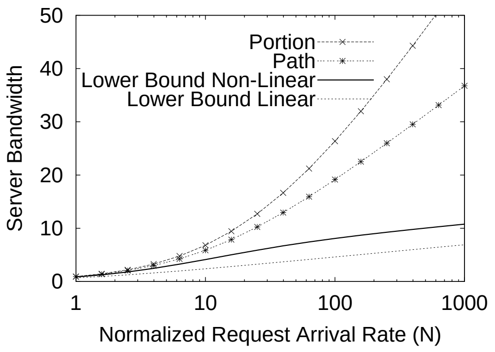


Figure 2: Server Bandwidth for Non-Linear Media (balanced binary tree with height 3, $\alpha = 1$, $d = 0$)

a priori knowledge of client path selection. In this case, a simple approach is to treat each portion i of the non-linear media file as a separate linear media file that is requested d_i time units before it is needed, with $d_1 = d$ and d_i, $i > 1$, less than or equal to d plus the sum of the playback time for all portions played prior to i. This yields a tight lower bound on required server bandwidth that is the sum of the lower bounds for delivering each linear portion, as follows:

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

For the results shown in Fig. 2, it is assumed that $d_i = 0$ for all i, which corresponds to immediate service and no early client path selection.

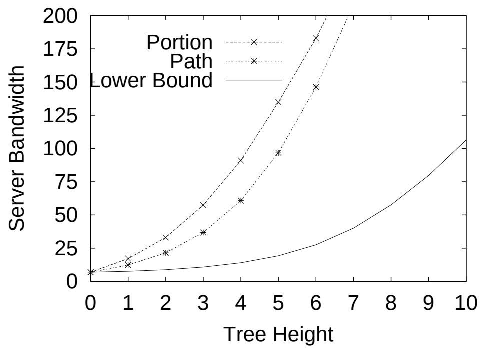
In the other approach (*path*), each client path selection is required to be known *a priori*, before the client receives the first video segment. In that case, one straightforward approach is to replicate the video data so that each complete path through the tree structure is stored as a separate file. For each client request, one of these files is selected with probability equal to the path selection probability, and delivered as if it were an ordinary linear media file. The corresponding tight lower bound on the required server bandwidth is given by

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

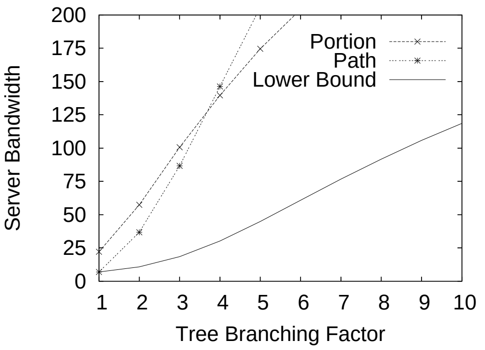
where $\mathcal{L}$ denotes the set of indices of the portions of the video file that are leaves in the tree structure, and where for notational convenience it is assumed that each complete path has the same playback time T.

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) fully exploiting this potential requires techniques that exploit the particular non-linear structure, rather than treating each portion or path as a separate linear media file.

The potential bandwidth reductions from multicast-based delivery are dependent on the non-linear media structure. Fig. 3(a) shows the impact of increasing the height of a balanced binary tree structure, for fixed normalized request rate ($N = 1000$) and immediate service ($d = 0$). As the



(a) Impact of Tree Height (balanced binary tree)



(b) Impact of Branching Factor (a tree height of 3)

Figure 3: Impact of the Non-Linear Media Structure ($\alpha = 1$, $N = 1000$, $d = 0$).

height increases, the number of portions of the video file increases exponentially, as does the number of possible paths. Furthermore, branch points become relatively more closely spaced, i.e., the length of each video portion decreases relative to the total length of a path. Not surprisingly, the potential bandwidth savings of multicast-based delivery decrease. However, the potential savings are still substantial even for trees of height ten. Note that the gap between the bandwidth requirements for *portion* and *path*, and the lower bound, increases with the tree height.

Fig. 3(b) shows the impact of increasing the branching factor at each branch point, for fixed tree height and request rate. The potential benefits of multicast-based delivery decrease somewhat as the branching factor increases, owing to the resulting increase in the number of paths. Note also that *path* becomes less efficient than *portion*. As the branching factor increases, the potential for sharing multicasts of data from the path files used in *path* decreases, but the potential for sharing multicasts from the file used in *portion* for the root of the tree, which all clients receive, is unaffected.

A key conclusion from these results is that even with more than a thousand possible paths (i.e., a tree height of 10 in Fig. 3(a) or a branching factor of 10 in Fig. 3(b)), multicast-based delivery still has the potential for an order-of-magnitude reduction in server bandwidth, assuming immediate service and normalized request rate greater than or equal to 1000. These potential bandwidth savings are largely due to the potential for shared delivery of the video

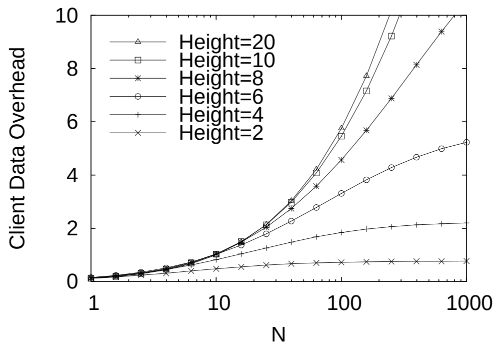


Figure 4: Client Data Overhead for Unrestricted Snoop-ahead (balanced binary tree, $\alpha = 1$, $d = 0$, no *a priori* knowledge of client path selection)

portions with the highest selection probabilities (i.e., those along popular paths or near or at the root).

3.3 Client Data Overhead for Unrestricted Snoop-ahead

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. We call this approach *unrestricted snoop-ahead*. Since data is multicast at minimum frequency to achieve the bound, it is guaranteed that the same data is not multicast more than once during the time that a client can obtain it. Thus, the average 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 listens to 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. Note that this is the *minimum* client data overhead that would be incurred with unrestricted snoop-ahead, since it assumes the minimum possible frequency of multicasts of data from each portion.

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, as computed using the above expression. Note that the average data overhead can be substantial. For example, in Fig. 4 it is greater than one (i.e., exceeds the amount of data on a complete path) when the tree height is at least 4 and N is at least 20.

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 $j > 1$. For fixed request rate, as the height increases the average client data overhead also initially increases but will level off and for $d = 0$ eventually decrease. The eventual decrease is due to the increase in the number of possible paths, which results in a small proportion of the multicasts that are of data from below a client's current play point in the media tree.

Similar observations can be drawn from experiments in which the branching factor at each branch point is varied, with fixed tree height. The data overhead initially increases with an increase in the request rate (for fixed average branching factor), or an increase of the branching factor (for fixed request rate), but eventually levels off.

4 Restricted Snoop-ahead

Due to client reception rate and/or buffer space limitations, the client data overhead 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 the overhead.

Snoop-ahead can be restricted in at least two basic ways. First, as considered in Section 4.1, restrictions may be based on distance from the current play point. Second, as considered in Section 4.2, 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. Performance comparisons presented in Section 4.3 motivate a hybrid approach that combines both types of restrictions, which is described in Section 4.4.

4.1 Distance-based Restricted Snoop-ahead

A simple approach that restricts snoop-ahead based on distance is to only listen to 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 (termed *allnext*), clients listen to 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

$$B_{min}^{allnext} = \int_0^{T_1} \frac{\mathsf{d}x}{d + x + \frac{1}{\lambda}} + \sum_{i=2}^{V} \int_0^{T_i} \frac{\mathsf{d}x}{T_{a(i)} + x + \frac{1}{\lambda_i}}$$

$$= \ln\left(\frac{N_1}{N_1\frac{d}{T_1} + 1} + 1\right) + \sum_{i=2}^{V} \ln\left(\frac{N_i}{N_i\frac{T_{a(i)}}{T_i} + 1} + 1\right),$$

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 listen to transmissions from portions at most k branch points ahead, for fixed $k > 1$.

4.2 Client Path Prediction Approaches

With skewed branch selection probabilities, it may be possible to substantially reduce the client data overhead, with a relatively modest cost in increased server bandwidth, by listening to multicast transmissions from only the most popular portion of the video following the next branch point. The corresponding tight lower bound for this approach (termed *popnext*) is given by

$$B_{min}^{popnext} = \int_0^{T_1} \frac{\mathsf{d}x}{d + x + \frac{1}{\lambda}} + \sum_{i \in \mathcal{P}} \int_0^{T_i} \frac{\mathsf{d}x}{T_{a(i)} + x + \frac{1}{\lambda_i}} + \sum_{i \in \overline{\mathcal{P}}} \int_0^{T_i} \frac{\mathsf{d}x}{x + \frac{1}{\lambda_i}}$$

$$= \ln\left(\frac{N_1}{N_1\frac{d}{T_1} + 1} + 1\right) + \sum_{i \in \mathcal{P}} \ln\left(\frac{N_i}{N_i\frac{T_{a(i)}}{T_i} + 1} + 1\right) + \sum_{i \in \overline{\mathcal{P}}} \ln\left(N_i + 1\right),$$

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 not the most popular, among their siblings, respectively (excluding the root portion, which has no siblings). Achieving this bound would incur an average client data overhead of

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

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

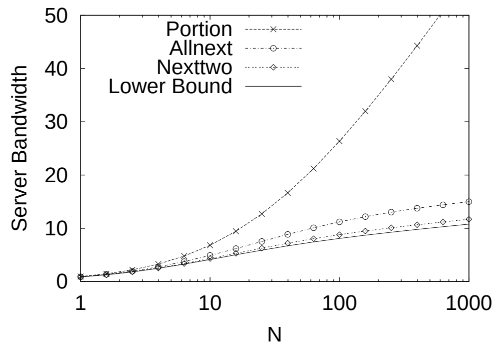
Instead of only listening to transmissions from the most popular video portion after the next branch point, clients could listen to transmissions from all video portions on the most popular path from the current position to a leaf. The corresponding tight lower bound for this approach (termed *poppath*) is given by

$$B_{min}^{poppath} = \int_0^{T_1} \frac{\mathsf{d}x}{d + x + \frac{1}{\lambda}} + \sum_{i \in \mathcal{P}} \int_0^{T_i} \frac{\mathsf{d}x}{\sum\limits_{j \in \mathcal{U}(i)} T_j + x + \frac{1}{\lambda_i}} + \sum_{i \in \overline{\mathcal{P}}} \int_0^{T_i} \frac{\mathsf{d}x}{x + \frac{1}{\lambda_i}}$$

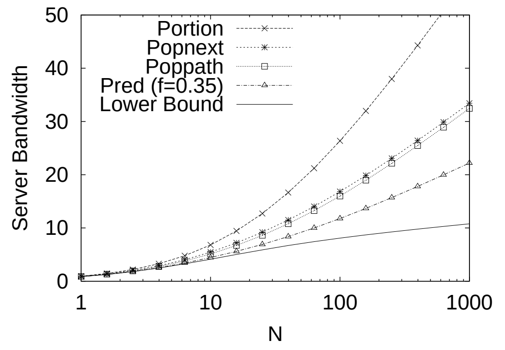
$$= \ln\left(\frac{N_1}{N_1\frac{d}{T_1} + 1} + 1\right) + \sum_{i \in \mathcal{P}} \ln\left(\frac{N_i}{N_i\frac{\sum_{j \in \mathcal{U}(i)} T_j}{T_i} + 1} + 1\right) + \sum_{i \in \overline{\mathcal{P}}} \ln\left(N_i + 1\right),$$

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

[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.



(a) Distance-based Policies



(b) Prediction-based Policies

Figure 5: Performance with Restricted Snoop-ahead (balanced binary tree with height 3, $\alpha = 1$, $d = 0$)

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

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

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 listen to 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, termed *pred*, 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 ($0 \le f \le 1$) are always successfully predicted, and less popular branch choices are never predicted. Note that perfect prediction is achieved for $f = 0$. As f increases, prediction accuracy decreases. For $f = 1$, no paths are ever predicted, and the approach has the same server bandwidth as *portion*. In the case of a binary tree structure, the approach is identical to *poppath* for $f = 0.5$ (assuming no two siblings with identical popularities). The tight lower bound on required server bandwidth for this approach is given by

$$B^{pred}_{min} = \int_0^{T_1}\frac{\mathsf{dx}}{d+x+\frac{1}{\lambda}} + \sum_{i\in\mathcal{F}}\int_0^{T_i}\frac{\mathsf{dx}}{\sum\limits_{j\in\mathcal{W}(i)}T_j+x+\frac{1}{\lambda_i}} + \sum_{i\in\overline{\mathcal{F}}}\int_0^{T_i}\frac{\mathsf{dx}}{x+\frac{1}{\lambda_i}}$$

$$= \ln\left(\frac{N_1}{N_1\frac{d}{T_1}+1}+1\right) + \sum_{i\in\mathcal{F}}\ln\left(\frac{N_i}{N_i\frac{\sum_{j\in\mathcal{W}(i)}T_j}{T_i}+1}+1\right) + \sum_{i\in\overline{\mathcal{F}}}\ln\left(N_i+1\right),$$

where $\mathcal{F}$ and $\overline{\mathcal{F}}$ denote the set of indices of those portions of the video file (excluding the root portion) whose conditional selection frequency is at least f, or less than f, respectively, and $\mathcal{W}(i)$ 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 in 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.) If an incorrect path prediction is for each of the paths that could have been predicted with probability proportional to its relative popularity, achieving this bound would incur an average client data overhead of

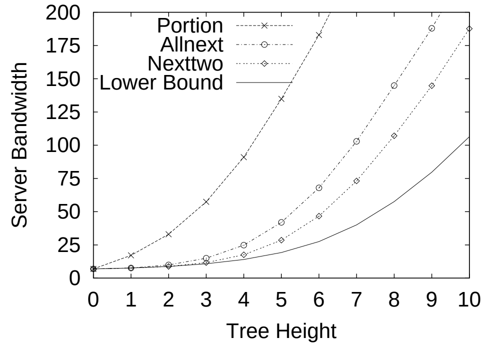
$$\frac{\sum\limits_{i\in\overline{\mathcal{F}}} p_i\left(\sum\limits_{j\in\mathcal{W}(i)} T_j\right)\sum\limits_{l\in\mathcal{L}(\mathcal{S}(i))}\frac{p_l}{\sum_{m\in\mathcal{L}(\mathcal{S}(i))}p_m}\sum\limits_{j\in\mathcal{D}(a(i),l)}B^{pred}_{j\ min}}{T}.$$

Here $\mathcal{L}(\mathcal{S}(i))$ denotes the set of indices of video portions that are leaves in the collection of pruned subtrees rooted at siblings of i, where the pruning has removed all video portions not in the set $\mathcal{F}$ and their descendents, $\mathcal{D}(a(i), l)$ denotes the set of indices of video portions on the path down to portion l beginning from (but not including) the parent of i, and $B^{pred}_{j\ min}$ denotes the bandwidth used for multicasts of portion j, as given by the term for portion j in the server bandwidth expression given above.

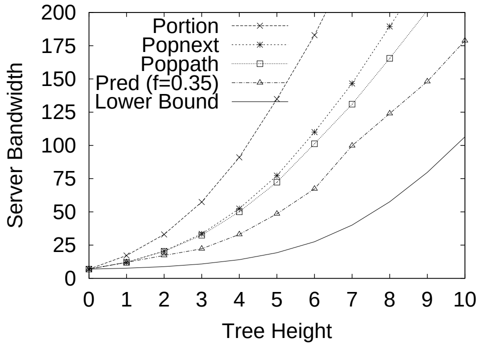
4.3 Performance Comparisons

Fig. 5 and Fig. 6 graph the bandwidth expressions derived in Sections 4.1 and 4.2 as functions of the request rate (for the balanced binary tree structure assumed for Fig. 2), and the tree height (for fixed request rate $N = 1000$), respectively. Also shown is the server bandwidth for the approach (*nexttwo*) in which clients snoop on multicasts from the current video portion plus from all portions following the next two branch points, which is derived similarly to that for *allnext*. For comparison purposes, the figures also show the server bandwidth for unrestricted snoop-ahead (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 Fig. 7 and Fig. 8.

Consider first the results for the distance-based approaches, *allnext* and *nexttwo*. In the *portion* approach, 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 (*allnext*) 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 minimum required server bandwidth fairly close to the



(a) Distance-based Policies



(b) Prediction-based Policies

Figure 6: Impact of Tree Height on Restricted Snoop-ahead Performance ($\alpha = 1$, $N = 1000$, $d = 0$)

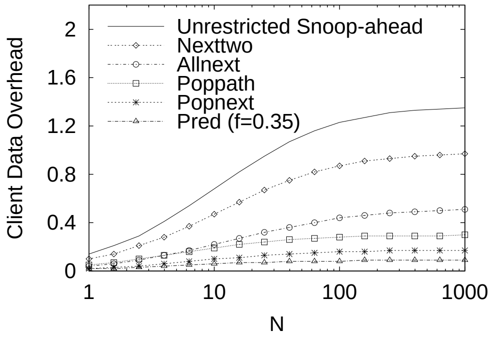


Figure 7: Client Overhead with Restricted Snoop-ahead (balanced binary tree with height 3, $\alpha = 1$, $d = 0$)

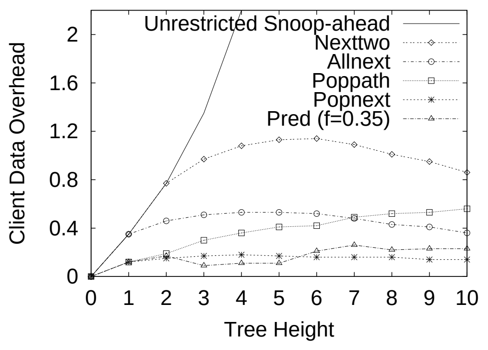
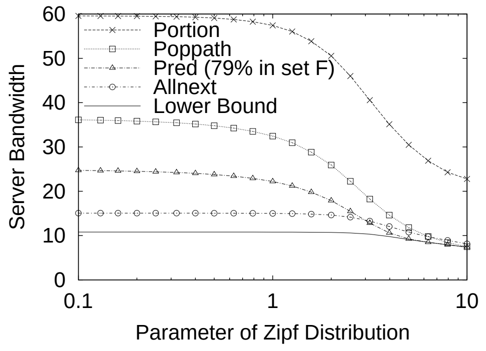
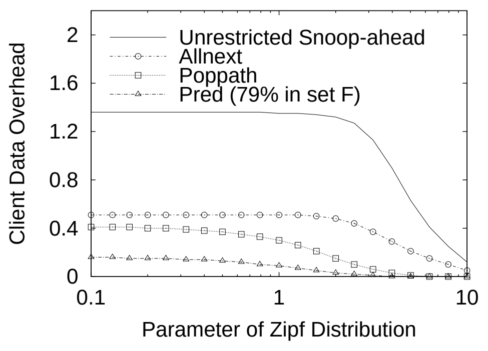


Figure 8: Impact of Tree Height on Overhead ($\alpha = 1$, $N = 1000$, $d = 0$)



(a) Server Bandwidth



(b) Client Overhead

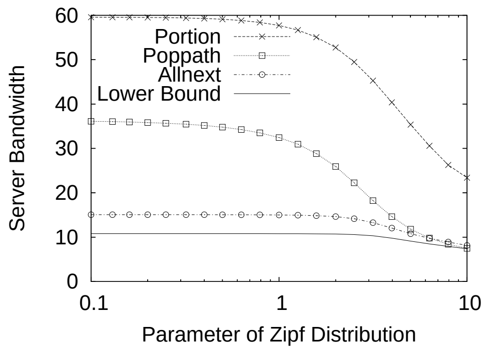
Figure 9: Sensitivity to Skewness in Selection Probabilities (balanced binary tree with height 3, $N = 1000$, $d = 0$)

lower bound of eq. 2. The results in Figs. 5(a), 6(a), 7, and 8 show that the *allnext* 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.

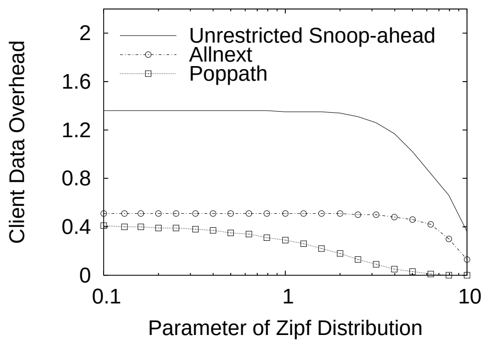
The *popnext*, *poppath*, and *pred (f=0.35)* approaches use *a priori* information regarding client path selection in an attempt to achieve a better tradeoff between server bandwidth and client overhead. Although *popnext* and *poppath* achieve low client overhead, as seen in Fig. 7 and Fig. 8, they achieve poorer server bandwidth scalability than *allnext* 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 *allnext*. In contrast, the more accurate *pred (f=0.35)* approach achieves lower client data overhead than *allnext* and comparable server bandwidth scaling.

Fig. 9 shows the sensitivity of the above results to the skew in the leaf selection probabilities, or more specifically to the value of the Zipf parameter α. Results for *nexttwo* and *popnext* are omitted for readability but have similar form. For *pred*, the parameter f varies with *alpha* such that the percentage of 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 *allnext* 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.

Fig. 10 shows the performance of several of the delivery approaches under an alternative path popularity model, described in Section 2.2, in which the selection frequencies at each branch point are Zipf-distributed. (For a binary tree, the selection probabilities at each branch point are $\dfrac{1}{1+1/2^\alpha}$ and $\dfrac{1/2^\alpha}{1+1/2^\alpha}$.) Results for *pred* are not shown since with this popularity model and the *pred* model of path prediction accuracy, it is possible to achieve only three points on the prediction accuracy spectrum (all branch choices successfully predicted, no branch choices successfully predicted, and 50% successfully predicted). Although this model yields a significantly different pattern of path popularities than the popularity model used for the previous results, the performance comparisons are very similar as can be seen by comparing Fig. 9 to Fig. 10.



(a) Server Bandwidth



(b) Client Overhead

Figure 10: Performance with Alternative Popularity Model (balanced binary tree with height 3, $N = 1000$, $d = 0$)

4.4 Hybrid Approach

The *pred* approach incurs a high bandwidth cost when a client takes a path that is not sufficiently popular to be successfully predicted. Motivated by the results in the previous section, this cost might be substantially reduced at relatively low cost in client data overhead by utilizing the *allnext* approach for such paths. Thus, we consider a hybrid approach (*hybrid*) that is similar to *pred*, except that all clients also listen to multicasts from all non-predictable video portions (i.e., those with conditional selection frequency less than the parameter f) immediately following the next branch point. Note that for $f = 1$, the *hybrid* approach becomes identical to *allnext*. The corresponding tight lower bound on the required server bandwidth for this approach is given by

$$\begin{aligned}
B^{hybrid}_{min} &= \int_0^{T_1} \frac{\mathsf{dx}}{d + x + \frac{1}{\lambda}} + \sum_{i\in\mathcal{F}} \int_0^{T_i} \frac{\mathsf{dx}}{\sum\limits_{j\in\mathcal{W}(i)} T_j + x + \frac{1}{\lambda_i}} \\
&\quad + \sum_{i\in\overline{\mathcal{F}}} \int_0^{T_i} \frac{\mathsf{dx}}{T_{a(i)} + x + \frac{1}{\lambda_i}} \\
&= \ln\left(\frac{N_1}{N_1 \frac{d}{T_1} + 1} + 1\right) + \sum_{i\in\mathcal{F}} \ln\left(\frac{N_i}{N_i \frac{\sum_{j\in\mathcal{W}(i)} T_j}{T_i} + 1} + 1\right) \\
&\quad + \sum_{i\in\overline{\mathcal{F}}} \ln\left(\frac{N_i}{N_i \frac{T_{a(i)}}{T_i} + 1} + 1\right).
\end{aligned}$$

Achieving this bound would incur an average client data overhead of

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

$$+\frac{\sum\limits_{i=2}^{V} p_i T_{a(i)} \sum\limits_{j\in\mathcal{S}(i)\cap\overline{\mathcal{F}}} \ln\left(\frac{N_i}{N_i \frac{T_{a(i)}}{T_i} + 1} + 1\right)}{T}.$$

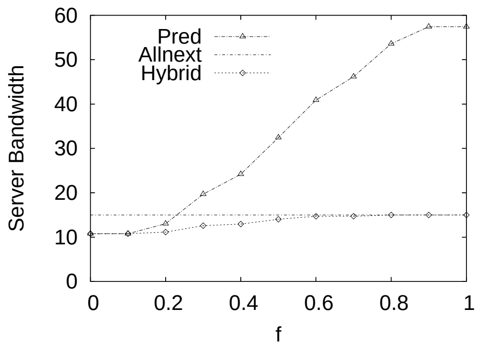
Fig. 11(a) and (b) show respectively the server bandwidth requirement and the client data overhead of the *hybrid* approach, as a function of f. In the region of most interest ($f < 0.5$), *hybrid* achieves a better trade-off between the required server bandwidth and the client data overhead than *allnext* and *pred*.

5 Scalable Delivery Protocols

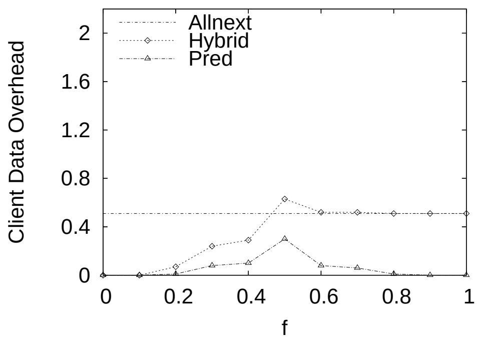
5.1 Hierarchical Stream Merging

Hierarchical stream merging (HSM) protocols [4–6], 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 have been merged may subsequently take different paths at a branch point, thus splitting the group. In that case, the server will need to start additional stream(s) so that there is one stream per path followed. Also, a more complex policy may



(a) Server Bandwidth



(b) Client Overhead

Figure 11: Performance of the *Hybrid* Approach (balanced binary tree with height 3, $\alpha = 1$, $N = 1000$, $d = 0$)

be required for determining what earlier stream a client listens to, in the case where the closest earlier stream is beyond the next branch point. Moreover, when a client or group of clients merges with an earlier group, the clients in the earlier group may restart listening to earlier stream(s) (as in HSM for linear media), or can alternatively continue snooping on the previous stream(s). In the latter case, clients in a single group listening to the same earlier stream may accomplish merges at different times.

Depending on how the above issues are handled, and on the availability of *a priori* path selection information, a number of non-linear media HSM variants can be defined. The two variants *HSM-Unknown Path* (HSM-UP) and *HSM-Popular Path* (HSM-POP) do not attempt client-specific path prediction. In HSM-UP, clients always listen 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). In HSM-POP, clients listen to the closest stream on the most popular branch if the closest earlier stream is beyond the next branch point. *HSM-Known Next* (HSM-KN) assumes that partial knowledge of client path selection, specifically the branch a client will select at the next branch point (but not yet the subsequent branch choices), can be known in advance. *HSM-Known Path* (HSM-KP) assumes that precise client-specific path prediction is possible. In both HSM-KN and HSM-KP, clients listen to the closest earlier stream delivering data from the known (partial) path they will select. Note that with these two protocols, clients belonging to the same group may be listening to different earlier streams.

An HSM protocol simulator was developed to assess the performance of the non-linear media HSM variants described above [18]. The simulator simulates the random arrival of client requests, the stream initiation that occurs at each client arrival, the merging and splitting of client groups according to the HSM protocol being simulated, the stream initiations that occur owing to the splitting of client groups at branch points, and the stream terminations when clients merge or when a stream reaches the end of the video. The input to the simulator is the client request rate, and the output is the server bandwidth used. Client reception and playback is not simulated, as this does not affect the required server bandwidth measure, assuming the client has sufficient buffer capacity. Previous work on HSM delivery of linear media has shown that client buffer sizes on the order of 10% of the file size are sufficient to achieve much of the performance gains of HSM delivery in this context [4]. The impact of limited client buffer capacity on required server bandwidth for the non-linear media delivery techniques is left for future work.

In the non-linear HSM simulations whose results are presented here, when a client or group of clients merges with an earlier group, all clients in the earlier group restart listening to earlier streams. It is also assumed that when a group splits at a branch point, the initiation of a stream for each new group is delayed until the playback position of the earliest client in that group reaches the branch point. Other options are investigated in [18].

Fig. 12 presents simulation results for the HSM-UP, HSM-POP, HSM-KN, and HSM-KP HSM variants described above, assuming Poisson request arrivals. HSM-UP has essentially the same required server bandwidth as HSM-POP, which is consistent with the result from Section 4 that using overall path selection probabilities to determine which multicast transmissions to receive may not be a fruitful strategy. On the other hand, precise advance knowledge of client path selection enables better choices of which earlier stream to listen to in HSM-KN and HSM-KP, thus substantially reducing their server bandwidth usage.

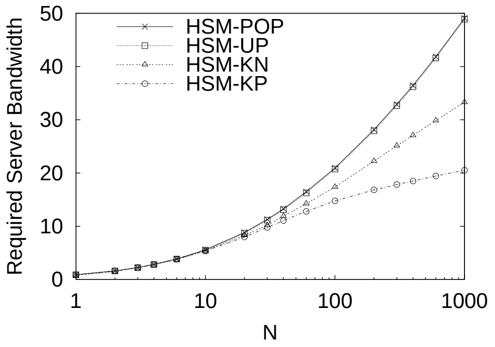


Figure 12: Server Bandwidth Requirements of Non-Linear HSM Variants (balanced binary tree with height 3, $\alpha = 1$)

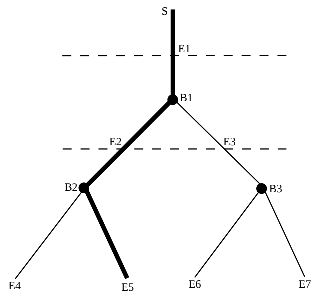


Figure 13: OPB-KP Segment Partitioning for an Example Media Structure ("S" label is for start of media, "Bi" labels are for branch points, "Ei" labels are for segment end points, dashed lines indicate segment boundaries, $K = 3$, $s = 2$, $r = 1$)

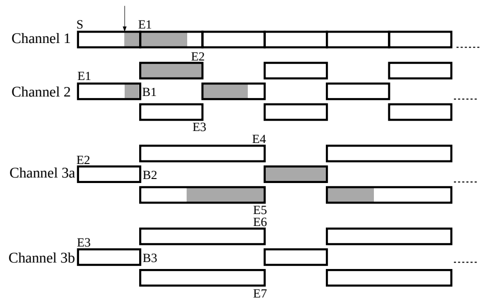


Figure 14: OPB-KP Channels for Structure of Fig. 13 (shaded areas are listening periods of example client, $K = 3$, $s = 2$, $r = 1$)

5.2 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 [13]. 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 s channels. The segment size progression is 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 *OPB-Known Path* (OPB-KP). Each complete path through the non-linear media file is partitioned using the same segment size progression as in OPB 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 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. 14 shows the channels used in the OPB-KP protocol for the example non-linear video structure shown in Fig. 13, 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 ($r = 1$). Note that in Fig. 13, each video portion is modeled by a segment in the tree rather than by a node, so that the partitioning of the video into broadcast segments can be shown. Fig. 14 also shows the periods during which an example client listens to the transmissions on each channel, assuming the client begins reception at the point indicated by the arrow and that the client takes the path shown in Fig. 13. As will be seen in Section 5.3, if the server can detect if there are any listeners on a channel (or sub-channel), and stop transmitting on the channel if not, this protocol 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. 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 suggested by its size in bytes (and corresponding transmission time), since the client will play 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. 16 shows the channels used in this *OPB-Unknown Path* (OPB-UP) protocol for the example non-linear video structure shown in Fig. 15, 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. 15.

Feasible segment sizes for OPB-UP can be computed using the algorithm outlined in Fig. 17. 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

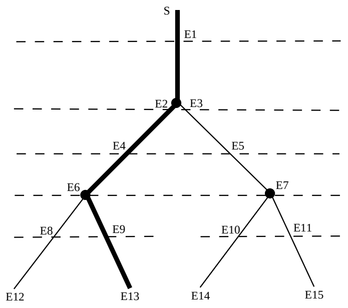


Figure 15: OPB-UP Segment Partitioning for an Example Media Structure (“S” label is for start of media, “Ei” labels are for segment end points, dashed lines indicate segment boundaries, $K = 6$, $s = 2$, $r = 1$)

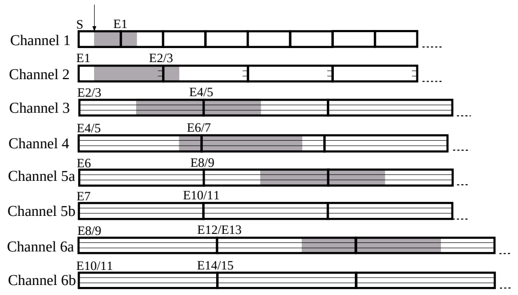


Figure 16: OPB-UP Channels for Structure of Fig. 15 (shaded areas are listening periods of example client, striped areas indicate multiplexed transmissions, $K = 6$, $s = 2$, $r = 1$)

time when a client begins reception of the segment, measured from the start of the video file playback, e_k denotes the latest time by which a client can end reception of the segment, measured from 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. 17 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.

For OPB, partial (but perfectly accurate) path prediction could be exploited using a hybrid of the OPB-KP and OPB-UP protocols. Errors in path prediction, however, would be difficult to recover from. 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.

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

Procedure $Compute\_Segsize(K, d, s, r)$
8.  $l_1 = dr$
9.  For ($k = 2$; $k \le K$; $k$++)
10.     $u_k = -d, \ k \le s;$
        $u_k = s$'th latest of $\{\ u_j + l_j/r \mid 1 \le j < k\ \}, \quad k \ge s+1$
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
```

Figure 17: Algorithm for OPB-UP Segment Sizes (balanced binary tree)

5.3 Performance Comparisons

Figs. 18, 19, and 20 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. The results for the OPB variants are computed analytically based on the channels used (determined as described in Section 5.2 by the parameters K, s, and r, and the media structure), and assuming that transmission on a channel is stopped whenever no client is listening to that channel. The fraction of time that no client is listening to a channel can be easily derived given Poisson request arrivals (also

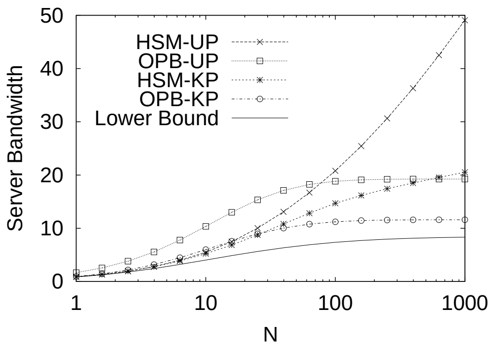


Figure 18: Performance of Scalable Delivery Protocols (balanced binary tree with height 3, $\alpha = 1$, $d = 0.01$ for OPB and lower bound, $r = 0.25$, $s = 8$)

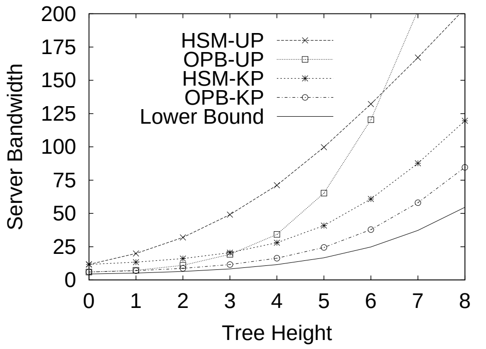


Figure 19: Performance with Varying Height ($\alpha = 1$, $N = 1000$, $d = 0.01$ for OPB and lower bound, $r = 0.25$, $s = 8$)

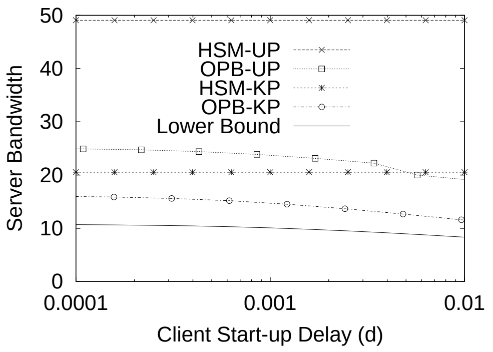


Figure 20: Impact of OPB Start-up Delay (balanced binary tree with height 3, $\alpha = 1$, $N = 1000$, $r = 0.25$, $s = 8$)

assumed in the simulations of the HSM protocols).

For HSM-KP and OPB-KP, path prediction is assumed to be perfect. For HSM, imperfect or partial path prediction (as exploited in HSM-KN) yields results intermediate to those for HSM-UP and HSM-KP, as shown in Fig. 12.

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 outperforms HSM-KP and yields performance as close to the lower bound from eq. 2 as could be expected, given that with $r \times s = 2$ each OPB-KP client receives data at a maximum aggregate rate of twice the streaming rate, whereas the lower bound places no such restriction (see [6, 13] regarding the impact of client receive rate limitations), and (4) the precise relative performance of the HSM and OPB variants that have incomplete knowledge of client prediction depends on the 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 [5]). The results for HSM suggest that it may be fruitful to investigate HSM variants in which clients can snoop on multiple earlier streams (e.g., one for each possible choice at the next branch point, similar to *allnext*), for use in contexts where accurate client specific path prediction is not possible.

5.4 Prototype Implementation

A rudimentary implementation of scalable non-linear media streaming has been added to the SWORD prototype streaming system [13]. The SWORD system consists of server and client components, each built using the open source Apache proxy server code as a basis. These interpose between Windows Media servers and players, and replace the normal unicast delivery with multicast delivery using hierarchical stream merging. Selective termination of streams by the server component and reconstruction of the data received on multiple multicast streams at the client component allow the use of hierarchical stream merging to be transparent to the client player.

Our implementation of non-linear media streaming stores each portion of the non-linear structure as a separate file. Modification of the header fields and spoofing of requests by the SWORD client component allow this non-linear structure to be transparent to the client player, to which it appears that only a single video file is being played (transitions between the video portions are seamless). These changes are implemented using approximately 500 lines of C^{++} code. For the SWORD server component, the main modification is the addition of code implementing the possible change in merge target (the earlier stream a client is listening to) when a client reaches a branch point and makes a selection. This addition is realized using about 200 lines of C^{++} code. 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.

6 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 significant potential for bandwidth reduction using multicast delivery in this context. The bounds also illuminate the advantages and 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 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. The new protocols achieve much of the potential bandwidth savings. Furthermore, 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 improved stream merging and periodic broadcast protocols, and further development and experimentation with the SWORD 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*, Hiroshima, Japan, June 1996.

[2] Y. Birk and R. Mondri, "Tailored transmissions for efficient near-video-on-demand service," in *Proc. IEEE ICMCS'99*, 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*, Las Vegas, NV, Sept. 1997.

[4] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM MULTIMEDIA'99*, Orlando, FL, Nov. 1999.

[5] ——, "Bandwidth skimming: A technique for cost-effective video-on-demand," in *Proc. IS&T/SPIE MMCN'00*, San Jose, CA, Jan. 2000.

[6] ——, "Minimizing bandwidth requirements for on-demand data delivery," *IEEE Trans. On Knowledge and Data Engineering*, vol. 13, no. 5, pp. 742–757, Sept./Oct. 2001.

[7] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proc. NOSSDAV'98*, Cambridge, UK, July 1998.

[8] L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in *Proc. ICMCS'99*, Florence, Italy, June 1999.

[9] A. Hu, "Video-on-demand broadcasting protocols: A comprehensive study," in *Proc. IEEE INFOCOM'01*, Anchorage, AL, Apr. 2001.

[10] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM MULTIMEDIA'98*, Bristol, U.K., Sept. 1998.

[11] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM'97*, Cannes, France, Sept. 1997.

[12] L. Juhn and L. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Trans. on Broadcasting*, vol. 43, no. 3, pp. 268–271, Sept. 1997.

[13] 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*, vol. 11, no. 2, pp. 195–209, Apr. 2003.

[14] 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*, Santa Clara, CA, June 2000.

[15] J.-F. Pâris, "A broadcasting protocol for compressed video," in *Proc. EUROMEDIA'99*, Munich, Germany, Apr. 1999.

[16] S. Sen, L. Gao, and D. Towsley, "Frame-based periodic broadcast and fundamental resource tradeoffs," Comp. Sci. Dept., U. Mass-Amherst, Tech. Rep. 99-78, 1999.

[17] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *ACM Multimedia Systems J.*, vol. 4, no. 3, pp. 197–208, Aug. 1996.

[18] Y. Zhao, "Scalable streaming of stored complex multimedia," Ph.D. Dissertation, University of Saskatchewan, July 2004.

[19] Y. Zhao, D. L. Eager, and M. K. Vernon, "Efficient delivery techniques for variable bit rate multimedia," in *Proc. MMCN'02*, San Jose, CA, Jan. 2002.

[20] ——, "Network bandwidth requirements for scalable on-demand streaming," in *Proc. IEEE INFOCOM'02*, New York, NY, June 2002.