Optimized Periodic Broadcast of Non-linear Media

Niklas Carlsson[¶] Anirban Mahanti[§] Zongpeng Li[‡] Derek Eager[¶]
[¶]Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
[§]Department of Computer Science and Engineering, IIT Delhi, India
[‡]Department of Computer Science, University of Calgary, Calgary, Canada

***Abstract*—Conventional video consists of a single sequence of video frames. During a client's playback period, frames are viewed sequentially from some specified starting point. The fixed frame ordering of conventional video enables efficient scheduled broadcast delivery, as well as efficient near on-demand delivery to large numbers of concurrent clients through use of periodic broadcast protocols in which the video file is segmented and transmitted on multiple channels.**

This paper considers the problem of devising scalable protocols for near on-demand delivery of "non-linear" media files whose content may have a tree or graph, rather than linear, structure. Such media allows personalization of the media playback according to individual client preferences. We formulate a mathematical model for determination of the optimal periodic broadcast protocol for non-linear media with piecewise-linear structures. Our objective function allows differing weights to be placed on the startup delays required for differing paths through the media. Studying a number of simple non-linear structures we provide insight into the characteristics of the optimal solution. For cases in which the cost of solving the optimization model is prohibitive, we propose and evaluate an efficient approximation algorithm.

I. Introduction

Conventional video consists of a single sequence of video frames that are viewed sequentially from a chosen starting point. Of interest in this paper is "non-linear" media consisting of multiple linear sequences seamlessly linked in a tree or graph structure, allowing multiple possible playback paths differing in the media portions they include and/or their ordering. One example is the generalization of the traditional linear movie to a non-linear form in which the viewer is able to choose among a variety of possible plot sequences and endings. Such non-linear formats are already available on DVD releases of movies (e.g., the DVD edition of the Hollywood movie "Clue" has three different endings). More conventional applications are evident in areas such as news-on-demand and virtual tours. For example, a customizable news on-demand service may allow clients to watch extended coverage of entertainment, sports, or political news following the coverage of the main headlines. Typically, in these customizable on-demand streaming services some of the content sequences are common to all viewers, with the remaining sequences shared among only a fraction of the media's viewers. As such, non-linear media offers benefits to both the media creator and the end user. For media developers, non-linear media offers the potential for creation of new media artifacts that would not be possible when constrained to the linear form. For clients, non-linear media allows *personalization* of the viewed content according to individual preferences.

The single linear frame sequence of conventional video has enabled use of efficient scheduled broadcast delivery systems (e.g., TV broadcasting), as well as development of scalable techniques for on-demand and near on-demand delivery. With on-demand delivery, incoming client requests for a media file are served immediately (system capacity permitting), allowing clients to view the media at times of their choosing. However, the total resource usage (server and/or peer, and network) increases with the rate of media file requests, although this increase can be reduced to sublinear (e.g., [3], [12]). Periodic broadcast protocols [5], [7], [9], [11], [14], [15] offer near on-demand service, in which each client incurs a playback startup delay of duration dependent on the server's transmission schedule, using fixed server bandwidth that is independent of the request rate. In the most efficient of these protocols, the required server bandwidth increases only logarithmically with linearly decreasing client startup delay [11].

One approach to delivery of non-linear media with piecewise-linear structures is to cyclically multicast each linear portion [4]. Like periodic broadcast, this approach can accommodate arbitrarily high client request rates using fixed server bandwidth; however, the required client startup delay is linear in the sizes of the linear portions. A related approach is to deliver each linear portion using a separate set of periodic broadcast channels, enabling the client startup delay to be decreased with logarithmic increase in server bandwidth; however, such approaches have been shown to be inherently less efficient than techniques that exploit the particular non-linear media structure, at least for some classes of media structures [16].

On-demand and near on-demand scalable delivery techniques that exploit knowledge of the possible or likely client paths through the media were proposed in [16]. However, the near on-demand periodic broadcast protocols were developed only for fairly narrow subclasses of media file topologies (e.g., trees). Furthermore, these protocols have the disadvantage of requiring the multiplexing of data from different paths on the same channel (implying that clients receive data that they do not require), and/or fragmented use of transmission resources using possibly many channels that cycle between active and idle states. An additional disadvantage is that the transmission schedules used in these protocols are heuristically determined. In contrast, for linear media, *optimized* periodic broadcast protocols have been developed that minimize client startup delay for a given server bandwidth allocation and client

To appear in *IEEE Transactions on Multimedia*. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

reception capacity [11], [14].

This paper concerns the problem of devising periodic broadcast protocols for non-linear media structures, that require neither the multiplexing of data from multiple paths on the same channel, nor fragmented use of transmission resources. Furthermore, of interest is the determination of *optimized* protocols that minimize a weighted average of path dependent client startup delays, for given server bandwidth allocation and client reception capacities.

We formulate a mathematical optimization model for periodic broadcast delivery of non-linear media structures that can be partitioned into a finite number of linear media segments, under the assumption that clients make their path choices (from a finite set of paths) at their arrival to the system. [1] A key component of our work is the modelling of periodic broadcast as a *linear optimization* problem, as first proposed for a context of linear media with heterogeneous clients [14]. The new optimization model for non-linear media files is substantially more complex than that for the linear media files. Specifically, our optimization model requires solution of a possibly very large number of linear programs (LPs) rather than just a single LP as in [14]. We develop solution space pruning methods, and exploit structure that we identify in the optimal solution for a particular class of scenarios, so as to substantially expand the range of cases for which exact solutions are feasible. For other cases in which the cost of solving the optimization model is prohibitive, we propose and evaluate an efficient approximation algorithm.

The remainder of the paper is organized as follows. Section II describes the class of periodic broadcast protocols of interest in this work. Section III presents our optimization model for non-linear media, the ways in which we expand the range of cases for which exact solutions of the model are feasible, and the approximation algorithm that we propose for use when the cost of an exact solution is prohibitive. Section IV presents numerical results illustrating the scalability properties of optimized periodic broadcast for non-linear media, the use of weights in the optimization model to obtain lower startup delays for particular classes of clients, and the accuracy and example application of the approximation algorithm. Conclusions are presented in Section V.

II. Periodic Broadcast

For scalable near on-demand delivery of linear media files, many periodic broadcast protocols have been proposed [5]–[7], [9], [11], [15]; a number of these are surveyed in [5]. Most periodic broadcast protocols have a similar structure. These protocols devote a fixed number of server channels per media file, and cyclically broadcast segments of the media file on these channels according to a predetermined schedule. Clients receive multiple segments concurrently at an aggregate rate that exceeds the media playback rate. Any data that is received ahead of when it is needed for playback is buffered. With appropriate design of segment lengths, channel transmission rates, and segment broadcast schedule, clients are able to receive all data in time for playback, with required server bandwidth that increases only logarithmically with decreasing client startup delay.

In this paper we consider the class of periodic broadcast protocols in which a (constant bit rate) media file is partitioned into K segments and each segment is repeatedly broadcast on a separate channel at a fixed rate r times the media playback rate. Clients download each segment of the media file in its entirety before playback of its media data begins. In addition to simplifying protocol design, this latter property is important for support of quality adaptation [14] and packet loss recovery [11].

Within the aforementioned class of protocols, optimized periodic broadcast (OPB) protocols have been developed [11]. The client download schedule and segment lengths in these protocols are optimized so as to minimize client startup delay for a given server bandwidth and client reception capacity. The protocols assume linear media and homogenous clients wherein each client can concurrently receive from s server channels. In the following, clients able to receive s server channels are said to have s "reception channels". Clients initially start downloading the first s segments. The download of each segment k, $k > s$, uses the same reception channel as that used for segment $k-s$ and begins as soon as the download of segment $k - s$ is complete; i.e., a round-robin, in-order, ordering of segment downloads across the reception channels is optimal in this context. (Note that the data for each segment is generally received out-of-order since a server channel may be at any point in its cyclic transmission of its segment when a client begins reception.) Clients may begin playback as soon as they have completed download of the first segment. The segment lengths are chosen such that clients continuing playback without pause will receive each subsequent segment just prior to this segment's playback point. Such segment lengths can be easily determined using equations that express the maximum length of each segment in terms of the lengths of the preceding segments. Subsequent work generalized the OPB protocols to the context of heterogeneous clients with differing reception capacities [14]. In these protocols, segment sizes are determined through solution of a linear program in which weights can be used to control the relative quality of service given to the various types of clients.

In addition to extensive analysis and simulation studies of the performance of periodic broadcast protocols, other work has shown the feasibility of such protocols using an Internet streaming video testbed [1]. While implementations of periodic broadcast protocols are most efficient in multicast enabled networks, [2] we note that cyclic multicast protocols (used to deliver each segment) can significantly reduce the resource usage at servers even in systems without multicast or broadcast channels [13].

[1]We note that 3D-graphic models used for video games (such as Second Life, for example) could be considered as non-linear media, but fall outside the scope of this paper. Such media is not composed of pre-existing "paths" or "segments" as in the type of media assumed here, and the users' navigation is dynamic rather than being fixed at the time the user enters the system.

[2]Whereas wide-area multicast is currently not implemented throughout the Internet, we note that there is much anecdotal evidence indicating that multicast is deployed within many sub-domains, private, and enterprise networks. Also, proxy-assisted multicast architectures [2], [8], [10] may be deployed by Content Distribution Networks.

III. Optimized Non-linear Broadcast

This section describes methods for obtaining optimized periodic broadcast protocols for near on-demand streaming of non-linear media. Section III-A describes our model and assumptions. Section III-B provides a mathematical formulation for the case in which clients are assumed to make their media selection at their arrival instance. This formulation allows the optimal channel allocation and segment size progression to be obtained by solving a large set of Linear Programs (LPs). Section III-C describes how the model can be applied to the case in which clients may defer their path selection decisions. The possible characteristics of optimal solutions are discussed in Section III-D. To expand the range over which optimal solutions are feasible, Sections III-E and III-F consider ways to reduce the number of LPs that must be solved; Section III-E introduces state space pruning methods, while Section III-F consider a class of scenarios for which the basic structure of an optimal solution can be determined. Finally, Section III-G describes an approximation algorithm that can be used when the cost of finding an optimal solution is prohibitive. For all algorithms the channel allocation and segment sizes progression can be calculated offline and their computational complexity therefore does not affect the performance of the clients. Table I summarizes notation.

TABLE I
Notation for Non-linear Broadcast

Symbol	Definition
K	Total number of server channels (segments)
r	Segment transmission rate (in units of media playback bit rate)
s_j	Number of channels that a client of type j can concurrently listen to
B	Server bandwidth
J	Total number of client types
L_e	Total media playback length of media portion e
n_e	Total number of server channels (and segments) allocated to media portion e
$l_{e,i}$	Total media playback length of segment i on media portion e
P_j	Set of all media portions along the path selected by client type j
E	Set of all media portions e in the media file ($E = \cup_{j=1}^{J} P_j$)
τ_j	Deterministic startup delay of clients of type j
w_j	Weight used for clients of type j
$t_j(k)$	Time by which a client of type j completes download of segment k along path selection P_j
$l_j(k)$	Total media playback length of segment k along path selection P_j
$prev_j(k)$	Index of the preceding segment received over the same reception channel as segment k along path selection P_j
t_m	Protocol threshold time
m	Protocol threshold index

A. Assumptions

As stated in Section II, this paper consider the class of periodic broadcast protocols in which a (constant bit rate) media file is partitioned into K segments and each segment is repeatedly broadcast on a separate channel at a fixed rate r times the media playback rate; thus the server bandwidth requirement $B = K \times r$. Clients download each segment of the media file in its entirety before playback of its media data begins.

We assume that the non-linear media file under consideration consists of a total of E linear media portions, each of length L_e. Further, each client type j ($1 \leq j \leq J$) is assumed to be associated with a path selection P_j, consisting of a sequence of $|P_j|$ media portions. Note that path selections may overlap, and furthermore, have different total media playback durations.

Fig. 1 illustrates simple non-linear media file structures. Fig. 1(a) represents a video with two alternative endings as a tree with height two. The edges of the tree represent the various media portions and the tree structure defines the order in which these portions may be received. Fig. 1(b) represents a case in which there are n possible beginnings, n different endings, and all clients share the middle portion.

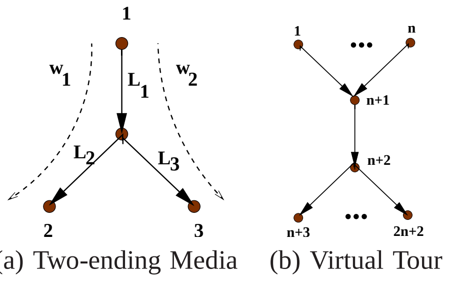


Fig. 1. Example Non-linear Media Structures

In addition to a path selection P_j, it is assumed that each client of type j has a reception bandwidth of $b_j = r \times s_j$, allowing it to concurrently download on s_j channels. We assume that the path selection P_j and client bandwidth b_j are known at the time of the client request. Given this information, the protocol provides each client with a startup delay τ_j and a schedule for receiving the required segments, which guarantees that each segment along a path is received by its playback time. Of interest is the problem of determining the segmentation and client download schedules which minimize the weighted startup delay over all clients $\sum_{j=1}^{J} w_j \tau_j$, where w_j is the weight given to the startup delay experienced by clients making path selection j. This weight could reflect, for example, the fraction of total requests for the media file that are generated by clients of type j, requesting path P_j. We consider only segmentations in which segments do not cross portion boundaries. Relaxing this restriction would yield protocols requiring the multiplexing of data from different paths on the same channel, and/or fragmented use of transmission resources, as in the protocols proposed in [16].

B. Mathematical Formulation

Under the assumptions described in Section III-A, the segment size progression for the Optimized Non-linear Broadcast protocol can be obtained by solving a large set of Linear Programs (LPs) and selecting the best solution. Each LP, with structure defined by constraints (3)-(8), is for a particular number of server channels (and segments) n_e allocated to each portion e, as well as a unique order in which clients of each type receive the segments of their path. The n_e values must satisfy the following constraints:

$$\sum_{e \in E} n_e = K \tag{1}$$

$$n_e \in N^+, \ \forall e \tag{2}$$

Note that there are a total of $\binom{K-1}{|E|-1}$ unique choices of the $\{n_e\}$ values that satisfy these constraints.

Further, given such a partitioning of the server bandwidth among the media portions, the segments needed to be retrieved along each path can be retrieved using many different client schedules. While a client selecting path P_j, may in some cases benefit from receiving the segments along this path out of order, it is never advantageous for the segments retrieved over the same reception channel to be retrieved out of order. Therefore, given a client with s_j reception channels, the $K_j = \sum_{e \in P_j} n_e$ segments along path P_j can be received according to $S(s_j, K_j)$ different schedules, where $S(s_j, K_j) = (1/s_j!) \sum_{i=0}^{s_j} (-1)^i \binom{k}{i} (s_j - i)^{K_j}$ is a Stirling number of the second kind. With J client types, each of the $\binom{K-1}{|E|-1}$ allocations of server channels to portions requires $\prod_{j=1}^{J} S(s_j, K_j)$ different LPs to be solved.

The inputs to each LP are the set of portions E in the non-linear media structure, their individual lengths (L_e's), the number of server channels allocated to each portion (n_e's), the number of server channels (K) and their transmission rate (r), the number of client classes (J) and their individual characteristics (s_j, P_j's) and weights (w_j's), as well as the set of segments received over each reception channel ($prev_j(k)$'s), for each of the J paths. Here, $prev_j(k)$ denotes the index of the preceding segment received over the same reception channel as segment k (and equals zero if there are no earlier segments scheduled on the same reception channel as used for segment k). The outputs from each LP are the media playback lengths of the segments ($l_{e,i}$'s), and the startup delay for each client type (τ_j).

To simplify the LP formulation, we let $t_j(k)$ denote the time by which a client of type j completes download of segment k, and $l_j(k)$ denote the k^{th} segment along path selection P_j, where segments are enumerated from 1 to K_j. In practice, for arbitrary non-linear media structures, a segment k and the previous segment received over the same channel $prev_j(k)$ can easily be obtained using a mapping between (j, k) and (e, i) pairs, as well as the channel each segment is received.

The LP for each configuration can be described as follows:

Minimize:

$$\sum_{j=1}^{J} w_j \tau_j \tag{3}$$

Subject to:

$$t_j(0) = 0, \qquad \forall j \tag{4}$$

$$t_j(k) = t_j(prev_j(k)) + \frac{l_j(k)}{r}, \quad \forall j, 1 \le k \le K_j \tag{5}$$

$$t_j(k) \le \tau_j + \sum_{k'=1}^{k-1} l_j(k'), \quad \forall j, 1 \le k \le K_j \tag{6}$$

$$\sum_{i=1}^{n_e} l_{e,i} = L_e, \qquad \forall e \tag{7}$$

$$l_{e,i}, \tau_j, t_j(k) \ge 0, \qquad \forall e, i, j, k \tag{8}$$

The objective function in equation (3) assigns a weight w_j to the startup delay experienced by clients making path selection P_j. Constraints (4) and (5) assume that clients having fully retrieved a segment immediately start downloading the next segment scheduled over the same reception channel. A client of type j listens to s_j concurrent channels, if possible, and is fully served when it has downloaded $K_j = \sum_{e \in P_j} n_e$ segments. Constraint (6) specifies that a client must always fully download segment k before completing playback of segment $k-1$. Further, the first segment must be downloaded prior to beginning of playback, that is within time τ_j of the client request. Constraint (7) ensures that the combined media duration of the segments along any media portion is equal to the media duration of that portion. Finally, constraint (8) ensures that quantities are non-negative.

C. Deferred Path Selection

While this paper focuses on the case in which clients are assumed to make their path selection at their arrival instance, alternative protocols can be designed which allow the clients to defer their path choices. This section describes how the model can be applied to the case in which clients may defer to some extent their path selection decisions until the time at which data from the next segment must start to be retrieved. Such protocols do not require clients to pre-fetch data from multiple future media portions in parallel, and in the case where the client reception bandwidth is not much greater than the play rate clients typically will be able to defer their path choices substantially. We note that protocols which require clients to retrieve data from multiple media portions in parallel result in much less efficient bandwidth usage [16].

Consider a client of type c, defined by the number of channels $s(c)$ it can receive in parallel and the first media portion it would like to obtain. Such a client may have multiple path choices; each path choice j is associated with the smallest possible startup delay τ_j. With at least one path choice being eliminated whenever a client picks a startup delay smaller than the largest startup delay required by any path selection with that same initial media portion, all path choices j with the same initial media portion must accommodate for the same startup delay. Assuming a client c is given a startup delay $\tau(c)$, this observation can be handled by adding the additional constraints that $\tau_j = \tau(c)$, whenever path choice j has the starting point that defines c. Finally, the objective function must be modified to $\sum w(c)\tau(c)$, where $w(c)$ is the weight given to clients of type c. All other constraints remain the same. Note that for the case in which the media structure corresponds to a tree and all clients have the same download capacity the objective function reduces to a single startup delay τ.

D. Characteristics of Optimal Solutions

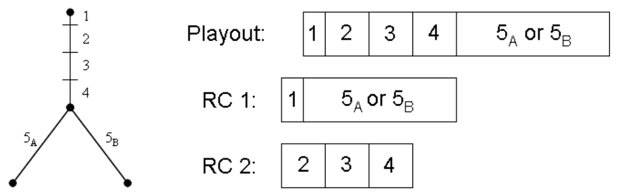
Optimal periodic broadcast protocols for non-linear media can differ greatly in their basic characteristics from those for linear media. [3] In particular, even for the context where clients have homogeneous reception capacities, optimal protocols may employ out-of-order segment retrieval, non round-robin retrieval of segments on the client reception channels, and (even for channel rates greater than the media playback rate) non-monotone segment length progressions.

Fig. 2 shows the optimal segment size progression and client download schedule on each of the two reception channels (RC1 and RC2) for a simple scenario with the same tree structure as previously described in Fig. 1(a). Here, each media portion is assumed to be of the same length, the server has a bandwidth of six times the playback rate, and the client has bandwidth to listen to two channels, each transmitting at the playback rate (i.e., $L_1 = L_2 = L_3$, $K = 6$, $s = 2$, $r = 1$). For this case it is optimal to allocate only a single server channel to each leaf portion (which is downloaded as a single segment) but to start downloading the respective leaf segment at the time of the first download completion. This schedule allows playback to begin at the download completion of the first segment. With the first segment being of half the size of all the other segments of the first media portion, this allocation results in a startup delay of 1/14 (measured in units of the total playback duration).

As shown in the figure, the optimal client schedule employs out-of-order retrieval and does not schedule the segments retrieved over the reception channels in round-robin order (i.e., every second segment is not scheduled on the same channel). Although segment lengths are monotonically non-decreasing in the above example scenario, in many cases this is not the case. For example, consider the media structure illustrated in Fig. 1(b). Clearly, when the length of the shared media portion becomes small the corresponding segment may become smaller than the segments used for the other media portions.

Another example with non-monotonically increasing segment sizes is a simple scenario in which clients select one of two paths, each consisting of the same media portions A and B, differing only in the order these media portions are played out. Assume both path choices are given equal weight ($w_{(A,B)} = w_{(B,A)}$), the length of the media portions are the same ($L_A = L_B$), $K = 6$, $s = 2$, and $r = 1$. In this case, it is optimal to allocate 3 channels to each media portion, and use the optimal segment size progression of linear media for each of the two portions individually. With this segment size progression the segment sizes of the second and third segment of each media portion are two and three

[3]See Section II for a description of the characteristics of optimal periodic broadcast of linear media.



(a) Server Allocation (b) Client Schedule

Fig. 2. Optimal Server Channel Allocation and Client Schedules ($L_1 = L_2 = L_3$, $K = 6$, $s = 2$, $r = 1$)

times the size of the first segment, resulting in a startup delay equal to 1/12 (measured in units of the playback duration). Thus, the segments retrieved by a peer are not monotonically non-decreasing in size. Also, this segment size progression causes any schedule for which the clients fully utilize their download channels until all segments are fully downloaded (and segments are scheduled back-to-back) to be constructed such that clients either retrieve segments for the second media portion out of order (e.g., clients start download of the sixth segment before beginning download of the fifth), or in non-round-robin order (e.g., clients download the fourth and fifth segment over the same channel).

Although optimal performance may not be achievable with round-robin schedules in some cases, perhaps in the cases of interest there is always a round-robin schedule that yields near-optimal performance. To investigate this question, relatively simple scenarios similar to the above tree (more specifically those discussed in Sections IV-A and IV-B) were considered. For each scenario an optimized version of the round-robin schedule (obtained using a modification of the solution techniques described in Section III-C) was compared with its corresponding optimal schedule (obtained using the techniques described in Sections III-C, III-D, and III-E). Among these 200 scenarios, 46 cases were found where the weighted startup delay with the best round-robin schedule was more than 5% higher than with the optimal schedule, 20 cases were found where the performance difference was more than 20%, and 5 cases were found where the performance difference was more than 50%.

E. Pruning Methods

Exhaustive search of the configuration space is typically infeasible. This section develops rules for pruning the search space. These rules are based on checking whether the weighted startup delay with the current best configuration is no greater than a lower bound with some set of alternative configurations, in which case the latter need not be considered further. In particular, we use a lower bound $\tau^{lb}(\{n_e\})$ on the weighted startup delay possible with a given allocation $\{n_e\}$ of server channels to portions.

Through the sharing of common media portions clients impact both the segment sizes along their own path, as well as those along the paths of other client types. The lower bound discussed in this section ignores such dependencies and determines the best possible segment size progression for each client type independently. More specifically, given an allocation $\{n_e\}$ the lower bound is calculated as:

$$\tau^{lb}(\{n_e\}) = \sum_{j=1}^{J} w_j \tau_j^{lb}(\{n_e\}), \quad (9)$$

where $\tau_j^{lb}(\{n_e\})$ denotes a lower bound on the startup delay of client type j when these clients are considered in isolation (i.e., the performance of other client types is given no consideration).

There are many possible ways to obtain a lower bound $\tau_j^{lb}(\{n_e\})$. For the special case where a path consists of a single media portion, optimal segment sizes (and thus a tight lower bound on client delay) can be obtained as described in [11]. Section III-E1 discusses how a lower bound can be obtained for a client type j with $s_j = 2$ (i.e., clients have two reception channels). Section III-E2 discusses more generally applicable bounds. With all bounds considering each client type j in isolation, the index j is omitted throughout the remainder of this section.

1) Lower Bound for Two Reception Channels: We first consider the case in which the path of the client type under consideration consists of two media portions of duration L_{e_1} and L_{e_2} with n_{e_1} and n_{e_2} segments, respectively. Similarly as for the case of a path with just a single media portion, the startup delay can be minimized by using maximally sized segments (for both media portions). However, there is sometimes an advantage to beginning reception of the second media portion before downloads of all segments of the first media portion have been initiated, resulting in out-of-order segment retrieval.

The client reception schedule is specified by giving the time at which each segment is "scheduled" (i.e., the time at which the client starts to download it) and the reception channel on which reception is scheduled. With maximally sized segments, any segments scheduled before the first segment of the second portion can be scheduled on alternating channels. Assuming the first segment of the second portion is scheduled at the completion of the m^{th} segment ($0 \le m < n_{e_1}$), the first m segments are scheduled in round-robin fashion while the next $n_{e_1} - m$ segments (belonging to the first media portion) are scheduled sequentially over the other channel (in parallel with the first segment of the second media portion). Finally, the n_{e_2} segments of the second portion are best scheduled in round-robin fashion (as this allows their sizes to be maximized). This yields:

$$l(k) = \begin{cases} (t(k) - t(m))r, & \text{if } k = n_{e_1} + 1 \\ (t(k) - t(k-1))r, & \text{if } m + 1 < k \le n_{e_1} \\ (t(k) - t(k-2))r, & \text{otherwise} \end{cases} \quad (10)$$

With minimum possible startup delay at least one segment must have a slack of zero, where the slack of a segment is defined as the difference between its playback time ($\tau + \sum_{k'=1}^{k-1} l(k')$) and its download completion time ($t(k)$). Hence the minimum startup delay (that allows all segments to be received in time for their individual playback) can be determined by $\tau = \max_k [t(k) - \sum_{k'=1}^{k-1} l(k')]$.

Note that the smallest startup delay τ_m^1 to satisfy all segments of the first media portion increases as the time $t(m)$ at which the client starts downloading data from the second media portion moves earlier, while the corresponding startup delay τ_m^2 to satisfy all segments of the second media portion decreases as $t(m)$ decreases. In the following discussion we assume that the completion time $t(m)$ of segment m (as well as the completion time of any earlier segment downloaded over the same reception channel as segment m) is constrained by a variable t_m, such that $t(m) \le t_m$. With maximally sized segments this restriction allows us to express the completion times as follows:

$$t(k) = \begin{cases} 0, & \text{if } k \le 0 \\ \min[t_m, \tau_m + \sum_{k'=1}^{k-1} l(k')], & \text{if } k \le m, \frac{m-k}{2} \in Z \\ \tau_m + \sum_{k'=1}^{k-1} l(k'), & \text{otherwise} \end{cases} \quad (11)$$

To find the optimal startup delay, all m ($0 \le m < n_{e_1}$) must be considered. For each possible value of m, the optimal startup delay τ_m can be obtained using a case-based algorithm. This algorithm first checks if either of the two special cases of $t_m = 0$ and $t_m = \infty$ provides an optimal solution. Let τ_m^0 and τ_m^∞ denote the smallest possible delay for each of these two cases.

First, consider the case of $t_m = \infty$. If this case is optimal, it is optimal to use maximally sized segments to deliver the first media portion, equalizing their slack to zero. Assuming this is the case, τ_m^∞ can be obtained by solving the equation system (10), (11) for the segments of the first media portion, with $t_m = \infty$ and the constraint that the sum of all segment lengths of the first portion must be L_{e_1}. This equation system can easily be solved in the same way as with linear media [11]. This startup delay τ_m^∞ is achievable if the value of m and the optimal segment size progression of the first media portion (most importantly $t(m)$ and $t(n_{e_1})$) allows the entire second portion to be retrieved in time of its individual segment playback times (i.e., using maximally sized segments for the second portion as necessary). If achievable, we consider the startup delay τ_m^∞ as a candidate solution.

Second, consider the case of $t_m = 0$ (implying $m = 0$ in the absence of zero sized segments). If this case is optimal, it is optimal to start retrieving the second media portion immediately at the client's arrival. Assuming this is the case, τ_m^0 can be obtained by solving the equation system (10), (11) for the segments of the second media portion, with $t(m) = 0$ and $t(n_{e_1}) = L_{e_1}/r$ and the constraint that the sum of all segment lengths of the second media portion must be L_{e_2}. This startup delay τ_m^0 is achievable if m and the optimal segment size progression of the second media portion allows each segment of the first portion to be retrieved in time of their individual playback times (i.e., using maximally sized segments using a single reception channel). If achievable, we consider the startup delay τ_m^0 as a candidate solution.

Finally, if neither of these special cases provide an optimal solution an intermediate value of t_m, provides an optimal solution. This corresponds to using maximally sized segments for all segments except those of the first media portion whose size is capped owing to the value of t_m. Such a solution can be obtained by solving the equation system (10), (11) with the additional constraints that:

$$\sum_{i=1}^{n_{e_1}} l_{e_1,i} = L_{e_1}, \sum_{i=1}^{n_{e_2}} l_{e_2,i} = L_{e_2}. \qquad (12)$$

Fig. 3 summarizes our algorithm. We note that this algorithm can be generalized to cases with more than two media portions. Here, it is always optimal for the first m_i ($0 \leq m_i \leq n_{e_i}$) segments of each media portion to be retrieved in round-robin order, with the remainder of the segments being retrieved sequentially. Only considering a single client schedule in isolation, the optimal segment order is found by computing the optimal segment size progression for each combination of m_i values, and then picking the solution with the smallest startup delay. Note that for a given set of m_i values the preceding segment $prev(k)$ can be determined.

```
for each $m = 0 .. (n_{e_1} - 1)$
    **if** solution $\tau_m^{\infty}$ is feasible for the second media portion
        $\tau_m = \tau_m^1$
    **else if** solution $\tau_m^0$ is feasible for the first media portion
        $\tau_m = \tau_m^2$
    **else**
        $\tau$ is the solution to (10),(11),(12) with $0 \leq t_m < \infty$
        $\tau_m = \tau$
$\tau_{min}$ =min$_m \tau_m$
```

Fig. 3. Single Path with Two Portions

In addition to using the above lower bounds when pruning possible server channel allocations, it is important to note that the above lower bounds on τ_m also can be used to prune certain combinations of client schedules from the set of client schedules considered for a given server channel allocation. In fact, most of our experiments use an inner pruning rule which prunes candidate LPs that correspond to a combination of client schedules that could not improve on the current candidate solution (even if the client types were considered independently).

2) General Case: As the proposed pruning approach can employ any valid lower bound $\tau_j^{lb}(\{n_e\})$ of the startup delay for a client type j, given a channel partitioning $\{n_e\}$, there are numerous other lower bounds that can be used. For example, consider allocating k segments to the first m media portions. Clearly, with segments not allowed to cross portion boundaries the achievable startup delay when ignoring portion boundaries (and considering the m media portions as a single composite portion) can never exceed the best achievable startup delay when considering these portion boundaries. Therefore, one possible bound is to consider the startup delay considering media only up until each portion boundary, when ignoring the preceding portion boundaries. While such bounds take the client reception bandwidth into account, tighter bounds can be obtained by taking all portion boundaries into consideration simultaneously. One brute force approach to obtain such (tight) lower bounds is to use LP formulations for each path (individually). While this may seem costly, it may in fact significantly reduce the search space (as there are many possible dependent client schedule combinations to consider for each valid channel partitioning). Here we use only pruning rules that do not require additional LPs to be solved.

It should further be noted that the effectiveness of all pruning rules significantly benefits from a good initial candidate solution, provided by the approximation algorithm described in Section III-F, for example.

F. Known Optimal Solution Structures

For the special case where $s_j = 2$ and $r \geq 1$, and the media has a tree structure, we conjecture that there exist optimal client reception schedules in which clients of type j:

- retrieve all segments of the respective leaf portion in round-robin fashion;
- retrieve some initial number of segments of the root portion in round-robin fashion and the remaining segments of the root portion sequentially (over the same reception channel); and
- for each intermediate portion, retrieve some initial number of segments sequentially (over the same reception channel), some additional number of segments in round-robin fashion, and the remaining segments sequentially (over the same reception channel).

The first of these properties can easily be shown because only peers taking path P_j will access the segments of this leaf portion. Therefore, the segment sizes can easily be optimized with respect to that client type alone, and the minimum number of segments needed to satisfy a given startup delay can always be achieved using maximally sized segments. Such schedules are by construction round-robin when $r \geq 1$. The second property is proven in the Appendix, while the third property is left as a conjecture.

Having proven the first two properties, we focus on a non-linear structure consisting of a single root portion (which all clients obtain) and a number of leaf portions (among which each client selects one). For such structures one out of n_{root} schedules is always optimal for each client type j, where n_{root} is the number of segments allocated to the root portion. These schedules are distinguished by which segment precedes the first segment of the second media portion; i.e., at which segment completion time a client skips ahead and starts downloading the first segment of the leaf portion. Assuming the first reception channel to complete download of its last segment from the root portion does so after having completely downloaded segment m_j ($0 \leq m_j < n_{root}$) the optimal schedule is determined as follows:

$$prev_j(k) = \begin{cases} 0, & \text{if } k = 1 \\ 0, & \text{if } k = 2, 1 \leq m_j \\ 0, & \text{if } k = n_{root}+1, m_j = 0 \\ m_j, & \text{if } k = n_{root}+1, m_j > 0 \\ k-1, & \text{if } m_j + 1 < k \leq n_{root} \\ k-2, & \text{otherwise} \end{cases} \qquad (13)$$

G. Approximation Algorithm

This section introduces a heuristic algorithm that can be used to effectively find server channel allocations, segment sizes, and client reception schedules that achieve near-optimal weighted startup delays for arbitrary non-linear media structures. This algorithm employs an outer loop that heuristically picks candidate allocations of server channels. For each such candidate allocation, another heuristic search is used to determine the segment download schedule that should be used by each client type. For each candidate configuration (consisting of a server channel allocation and a set of client download schedules) the optimal weighted startup delay and segment lengths are obtained by solving an LP. The following subsections describe each of these search heuristics. The accuracy and performance of the approximation algorithm are discussed in Section IV-C and Section IV-D.

1) Server Bandwidth Allocation Heuristic: As noted in Section III-C, there are $\binom{K-1}{|E|-1}$ possible server channel allocations. This number quickly becomes very large as the numbers of channels and portions increase. We employ local search heuristics to reduce the search space, as shown in Fig. 4.

```
(1) Initialize vector $n^*$ (i.e., choose initial channel allocation)
(2) Search client schedules (as described in Section III-G2)
      with fixed vector $n^*$
      Let $\tau^*$ denote the optimal weighted delay of best candidate
(3) $\forall e$: search client schedules (as in Section III-G2)
          with $n \leftarrow n^*, n_e \leftarrow n_e^* + 1$
      Let $\tau_e^+$ denote the optimal weighted delay of best candidate
(4) $e_+ \leftarrow \text{argmax}_e(\tau^* - \tau_e^+)$
(5) $\forall e$: search client schedules (as in Section III-G2)
          with $n \leftarrow n^*, n_e \leftarrow n_e^* - 1$
      Let $\tau_e^-$ denote the optimal weighted delay of best candidate
(6) $e_- \leftarrow \text{argmin}_e(\tau_e^- - \tau^*)$
(7) Search client schedules (as in Section III-G2)
      with $n \leftarrow n^*, n_{e_+} \leftarrow n_{e_+} + 1$, $n_{e_-} \leftarrow n_{e_-} - 1$
      Let $\tau'$ denote the optimal weighted delay of best candidate
(8) **if** $\tau' < \tau^*$ **then** $\tau^* \leftarrow \tau', n^* \leftarrow n$; Goto (3)
(9) $\forall e_+, e_-$: search client schedules (as in Section III-G2)
                  with $n \leftarrow n^*, n_{e_+} \leftarrow n_{e_+} + 1$, $n_{e_-} \leftarrow n_{e_-} - 1$
      Let $\tau'$ denote the optimal weighted delay of best candidate
(10) **if** $\tau' < \tau^*$ **then** $\tau^* \leftarrow \tau', n^* \leftarrow n$; Goto (3)
(11) output candidate ($^*$) and terminate
```

Fig. 4. Guided Local Search Algorithm

During each iteration of the algorithm in Fig. 4, a portion is identified for which the greatest improvement in weighted client startup delay is obtained when the portion is allocated an additional server channel, and a second portion is identified for which the least inflation of the weighted client startup delay is obtained when this portion is allocated one fewer channel. A "neighbor" allocation is obtained by switching one server channel between these two portions. If such a neighbor improves over the current candidate solution, the candidate solution is replaced and the localized search resumes. Otherwise, all possible neighbor allocations that can be obtained by the switching of a single channel are considered. If the best such neighbor improves on the current candidate solution, the candidate solution is replaced and the localized search resumes; otherwise, the algorithm terminates.

While any valid vector n can be used to initialize the search, the number of candidate solutions the guided local search algorithm must consider can be significantly reduced by using a more promising starting vector. For the numerical results presented in Section IV, we use a greedy extension of the optimized periodic broadcast protocol developed in prior work [11] for linear media files.

2) Client Scheduling Heuristic: As previously discussed, for each possible server channel allocation, there are a large number of client schedules that potentially could provide optimal solutions. This section proposes a search heuristic that significantly reduces the number of schedules that are considered.

The schedules considered include schedules with a restricted form of out-of-order segment retrieval, in which a reception channel may "skip ahead" to a later portion of the media file, even though there are one or more segments from the current portion that have not yet begun download. For each client type, however, we consider only schedules such that: (1) the segments received on each client reception channel are received in the order in which they occur in the client path (as would be true in any optimal schedule); (2) the time from the beginning of the first reception of a portion's segments until the last such reception can be divided into two periods, the first during which the set of reception channels downloading segments of the portion is added to over time (with none of these channels "skipping ahead" to a later portion), and during the second of which deletions occur to this set as channels move on to subsequent portions; (3) the segments of each portion are allocated in round-robin order, beginning with the first segment of the portion, to the (time varying) set of reception channels receiving segments from that portion; (4) a reception channel c may "skip ahead" from some portion e to a later portion, only if all channels that began reception of a segment from e earlier than c have already skipped ahead to e or to an even later portion; and (5) the order in which reception channels that do not "skip ahead" complete receiving data from a portion that is not the end portion of the path, and move on to the next portion, is the same as the ordering of the last segments they downloaded.

Given this class of client schedules, our search heuristic starts with an initial guess of promising segment schedules and then perturbs the schedules until no improvements are possible. Such an initial guess can either be obtained using a pure round-robin schedule for each client type or by using similar schedules to the client schedules used in the most promising candidate solution obtained so far. In each iteration, the algorithm attempts to improve the schedule of every client type one-by-one. For each client type, the algorithm considers neighboring schedules that allow one reception channel to be somewhat more aggressive or conservative. With the above class of schedules, there are at most $2s$ (typically less) ways of making the schedule for each portion more or less aggressive, respectively. Starting at the first media portion the algorithm considers neighboring schedules until the schedule either allows for an improvement in the weighted startup delay, or achieves the same weighted startup delay using a schedule that allows the download of later media portions to resume earlier (i.e., is a more aggressive schedule). If no changes are made to the schedule for any of the client types the search is terminated; otherwise, the search continues.

IV. Numerical Results

This section presents our numerical results. Without loss of generality, we measure bandwidth in units of the media playback bit rate, and normalize startup delays by the total playback duration of the selected path. For example, a startup delay of 0.01 means that the delay until playback can begin is 0.01 times the path playback duration. Numerical results are presented here only for scenarios in which the paths of all client types have equal duration.

Section IV-A considers the scalability of optimized broadcast protocols, while Section IV-B focuses on the impact of the weights. Throughout both these sections only media structures for which exact optimal solutions can be obtained are considered. Section IV-C evaluates the accuracy of the approximation algorithm described in Section III-G. Section IV-D illustrates use of the approximation algorithm for more general scenarios.

A. Scalability

We first consider the performance with the simple tree structure illustrated in Fig. 1(a) and $s = 2$. To obtain the optimal solution we use both the pruning rules defined in Section III-E and (when $r \geq 1$) the simplifying characteristics observed in Section III-F. Again, we note that the effectiveness of the pruning approach is highly affected by the order in which candidate solutions are considered and a good initial candidate is highly beneficial. For our numerical experiments we initialize our search using a greedy extension of the optimized periodic broadcast protocol developed in prior work [11] for linear media files. For example, for a variation of the structure shown in Fig. 1(a) with five branches rather than two, this approach requires 39 and 70 LPs to be solved when $B = 20$ and $B = 40$; in contrast, an exhaustive search require 11,628 and 575,757 allocations of server channels to be considered (and a much larger number of LPs to be solved), respectively. While the number of LPs is roughly the same for these two values of B, we note that the larger example requires much longer processing time as the number of channel allocations (and service schedules) that must be pruned is much greater for this case.

Fig. 5 shows that linear increases in server bandwidth result in exponential decreases in startup delay. This is a characteristic property which previously had been observed for periodic broadcast protocols delivering linear media files (e.g., [11]). Note that for cases with $r < 1$ we are limited to scenarios with smaller numbers of server channels. Fig. 6 shows how the size of the initial shared media portion impacts the startup delay and the amount of server resources allocated for this initial root portion. With the exception of the case where the performance is entirely constrained by the root portion and each leaf portion is assigned only a single server channel, the startup delay increases roughly exponentially with the percentage of the media file that is not shared among the clients. It should, however, be noted that the startup delay sometimes decreases within regions for which the same channel allocation is optimal. Further, because of increasing segment sizes, the root portion typically requires more server resources per unit of data served than the other media portions. Finally, Fig. 7 shows that the startup delay increases exponentially as the branching factor is increased, increasing the number of possible paths. The flattening of the lines for $B = 10$ and $B = 15$ corresponds to a region in the parameter space in which each leaf portion is allocated only a single channel and the leaf portions become the constraining portions. Reducing the number of channels allocated to the initial media portion therefore has a small effect on the startup delays.

B. Impact of Weights

Fig. 8 shows how the path weights influence the optimal periodic broadcast schedule of the non-linear media file. Again, the results are for the media structure shown in Fig. 1(a) with $s = 2$; the length of the non-shared portion of each path is chosen to be equal to 80% of the total file data along that path and the relative weight ratio w_1/w_2 is varied three orders of magnitude. Fig. 8(a) shows the startup delay as a function of the relative weight given to each path selection for two scenarios (with a server bandwidth of $B = 12$ and $B = 16$, respectively). Fig. 8(b) shows the number of channels allocated for each media portion for the case of $B = 12$. As expected, typically the leaf portion associated with a client type given very small weight is allocated only a single channel, while the other leaf portion is allocated significantly more server channels. As the weight given to the less weighted path increases the number of server channels allocated to each leaf portion becomes more balanced. The observed abrupt changes in startup delay result from changes in the number of server channels allocated to each portion.

While we omit results for different server channel rates (r) and for different ratios of shared and non-shared media, our results show that the impact of the weights is larger, and has more intermediate solutions, when the shared portion is small relative to the non-shared portion. This is because these scenarios allow more server channels to shift from the low weight path to the high weight path.

C. Accuracy of the Approximation Algorithm

To quantify the accuracy of the approximation algorithm the startup delays of the solutions obtained using the approximation algorithm were compared with the optimal solutions. This section considers the scenarios discussed in Section IV-A and IV-B, including the omitted experiments in Section IV-B (with different client reception rates as well as a few experiments in which the root portion is of the same size as the leaf portion). Out of these 200 scenarios, the approximation algorithm only failed to find the optimal solution in 3 cases. These cases have increased weighted client startup delay of roughly 5%, 11%, and 18%, respectively. Looking more closely at each of these three cases, the approximation algorithm gets stuck in a local minimum whenever the localized search heuristic fails to find neighboring allocations which provide

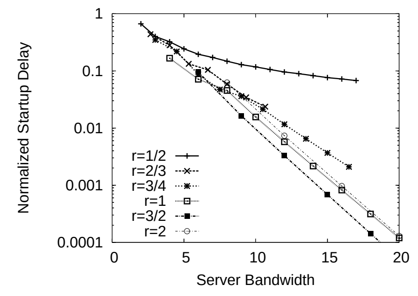


Fig. 5. Scalability: Server Bandwidth (media in Fig. 1(a); $L_1 = L_2 = L_3$, $w_1 = w_2$, $s = 2$)

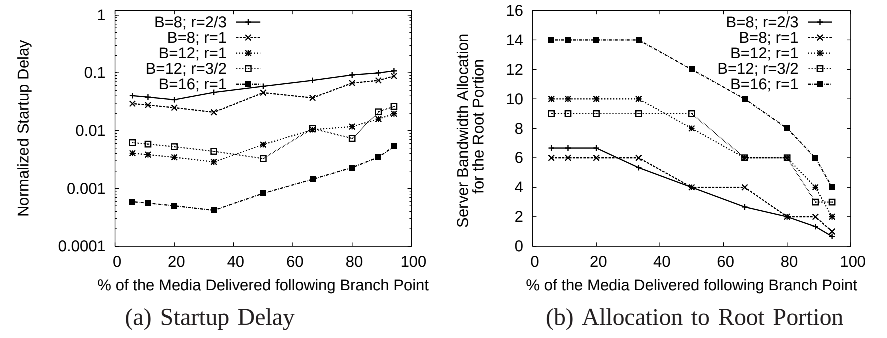


Fig. 6. Scalability: Impact of Shared Media Portion (media in Fig. 1(a); $L_2 = L_3$, $w_1 = w_2$, $s = 2$)

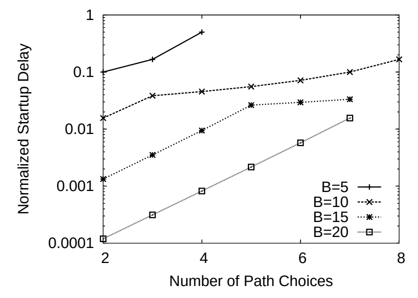


Fig. 7. Scalability: Number of Path Choices (media variants on Fig. 1(a); $L_1 = L_2 = ... = L_{J+1}$, $w_1 = w_2 = ... = w_J$, $s = 2$, $r = 1$)

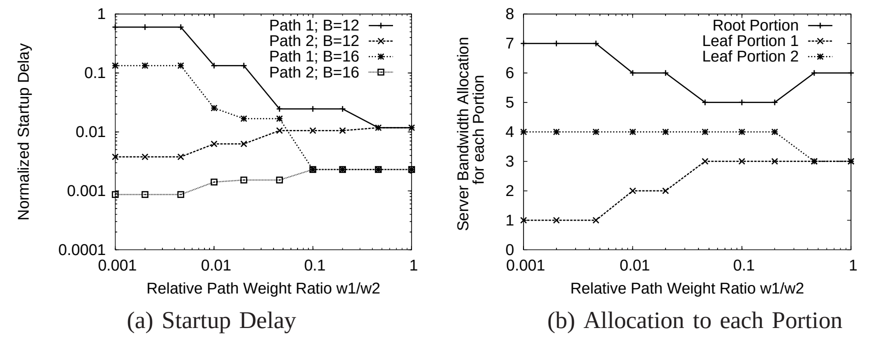


Fig. 8. Impact of Weights (media in Fig. 1(a); $L_2 = L_3 = 4L_1$, $s = 2$, $r = 1$)

improved performance (without moving multiple segments simultaneously).

While the fraction of scenarios for which the algorithm fails to find optimal solutions may be different for other structures, scenarios, and/or initialization algorithms, we do not expect the relative increase in startup delay (with respect to optimal) to become worse as the size of these basic structures increases. In fact, assuming a reasonable initial channel allocation, larger structures may reduce the relative increase in startup delay. While larger structures have more states (which potentially correspond to local minimums) we note that these states typically have many more neighboring states that may help the algorithm progress towards the optimal solution.

D. Performance on More General Non-linear Media Structures

The remainder of this section uses the approximation algorithm to consider a number of example scenarios for which the current pruning rules do not allow us to obtain exact optimal solutions. Fig. 9 shows the impact of client heterogeneity. We consider a scenario in which there are low and high reception rate clients, and where clients select a path consisting of the root portion and a different leaf portion of the media structure illustrated in Fig. 1(a), with all media portions of the same duration, yielding four client types. The server has a bandwidth of $B = 10$, with $K = 20$ and $r = 0.5$. The figure shows results for low and high reception rate clients having bandwidths of 1.5 ($s = 3$) and 2 ($s = 4$), as well as 1 ($s = 2$) and 4 ($s = 8$). The startup delay of each of the two client types with the same reception rate is given the same optimization weight. Note that when a small weight is given to high bandwidth clients the startup delay is the same for both client types. However, with more weight given to high bandwidth clients, the performance of high bandwidth clients is improved, at the expense of low bandwidth clients.

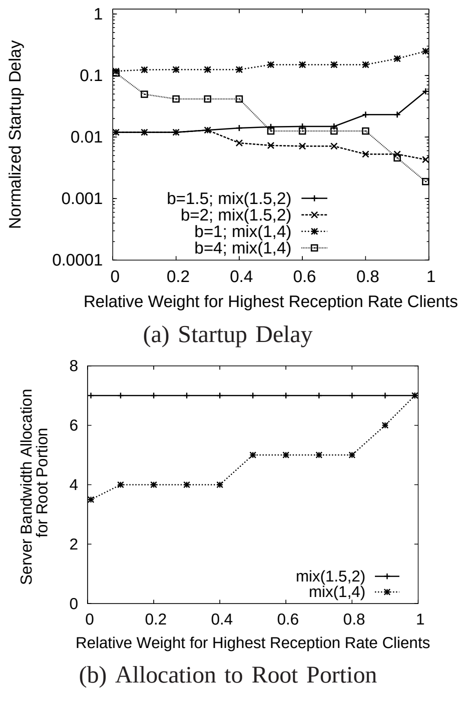
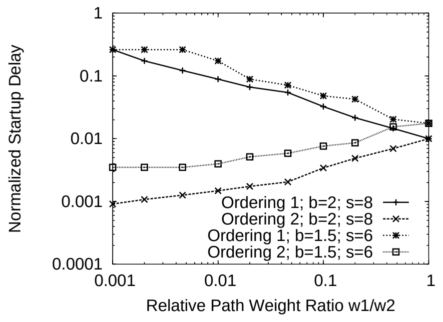
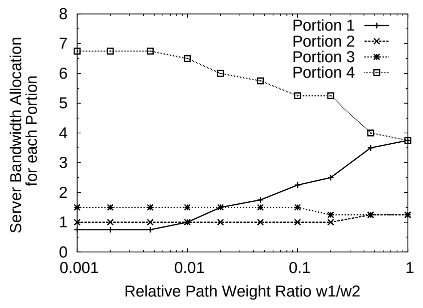


Fig. 9. Client Heterogeneity (media in Fig. 1(a); $L_1 = L_2 = L_3$, $B = 10$, $r = 0.5$)

As a second example, we consider a scenario in which two client types access the same four media portions, but in reverse order of each other. This could correspond to a customized newscast, for example. Fig. 10(a) shows how the startup delay of each client type is impacted by its relative weight, for two different example cases. In the first case both client types have a bandwidth of 2, while in the second case both client types have a bandwidth of 1.5. Note the strong impact the weights have on the startup delay. Fig. 10(b) shows the number of segments allocated to each media portion for the second example case. As expected, more channels are typically allocated to the media portion at the beginning of a client type's path. Further, as illustrated by the large differences in the server bandwidth allocated to portion 1 and 4, and between that allocated to portion 2 and 3, the number of channels allocated to each media portion is strongly skewed in favor of the client type given a larger weight.



(a) Startup Delay



(b) Allocation to each Portion

Fig. 10. Two Directional Media Structure with Four Media Portions ($L_1 = L_2 = L_3 = L_4$, $B = 10$, $r = 0.25$)

Similar to Fig. 7, Figs. 11-13 illustrate the impact of allowing more path selections, using only a fixed set of resources. The media files used for these experiments correspond to (i) a symmetric binary tree with a common initial portion, (ii) a symmetric tree of height three with a common initial portion, and (iii) the topology illustrated in Fig. 1(b) with clients of type j selecting the j^{th} initial as well as the j^{th} final portion. In all cases, all portions are the same duration. Although the scales are different, we note that the first data point in each figure corresponds to having a single path selection and the last data point having 32 different path selections.

To handle large media structures we again use the approximation algorithm described in Section III-G. This radically reduces the number of LPs that must be solved. For example, when $B = 12.5$, $r = 1/8$ (implying $K = 100$), and $s = 16$, for a binary tree of height 5 (with 31 edges) an exhaustive search requires $2 \cdot 10^{25}$ different channel allocations to be considered (and a much higher number of LPs to be solved). Using the localized guided search algorithm the number of channel allocations is reduced to 10^3, and the number of LPs to $3 \cdot 10^6$, where each LP consists of 10^3 constraints. [4]

When discussing the allocation for larger structures it should also be noted that, for scenarios in which clients have a download bandwidth that exceeds the play rate, the constraining portions will typically be located close to the starting point(s) and portions further away typically will be much less constrained. Therefore, the performance of the protocol is typically dependent on how well the initial portions are allocated channels. Further, we expect the convergence times of the search algorithm to be faster the more skewed the optimal channel allocation is (e.g., in scenarios where clients have much larger client bandwidth than the play rate).

Comparing Figs. 11-13, we note that the startup delay degrades much faster if additional path options are added to the beginning of the file than if such options are added later in the file. This is a consequence of the fact that earlier media portions typically require more server resources.

Whereas the full approximation algorithm significantly reduces the number of LPs that must be solved, we note that modifications to this algorithm easily can be applied to further reduce the number of LPs that need to be solved. This may be desirable for larger structures for which the number of constraints in each LP formulation is large. Table II illustrates the performance results using a modification in which the client schedule search heuristic only is applied when the localized guided search reaches a minimum; otherwise, the localized guided search does not invoke the client scheduling heuristic (and instead solves a single LP for each neighboring candidate allocation).

For a set of random non-linear media structures, Table II presents the startup delay, the number of constraints in each LP formulation, and the number of LPs that need to be solved using the above modified heuristic. [5] In this example, a server with bandwidth $B = 250$, $K = 250$, and $r = 1$ is delivering a non-linear media file consisting of $|E| = 100$ equally-sized media portions. Clients have $s = 2$ and randomly select one out of $J = 50$ path choices; each given equal weight $w_j = 1/J$. Each path consists of a random sequence of q media portions and has a playback duration equal to one (implying that the length of each media portion is $L_e = 1/q$).

For all cases considered, our modified approximation algorithm always (as desired) finds a solution in which the minimum of one channel is allocated to any portion that is not part of any of the J paths. While the random structures considered here are more complex than those considered

[4]If using an initialization in which all portions have as even a number of channels allocated to them as possible (rather than our regular initialization vector) the number of LPs that must be solved increases by a factor of 2. While this factor is highly variable it appears that this factor remains within a factor 5 for the experiments presented in Fig. 11 and typically obtains a solution with the same startup delay, suggesting that a good initial vector is beneficial, but not crucial, to the performance of our approximation algorithm.

[5]Due to details of the solver, the number of constraints reported here does not include variable definitions such as equation (2) and (8).

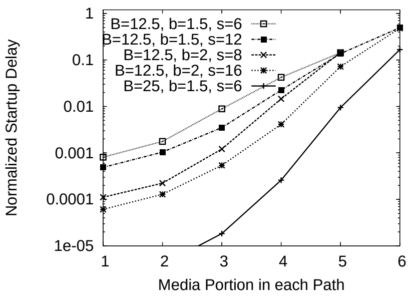


Fig. 11. Impact of Tree Height (binary tree with a common initial media portion; all portions the same size)

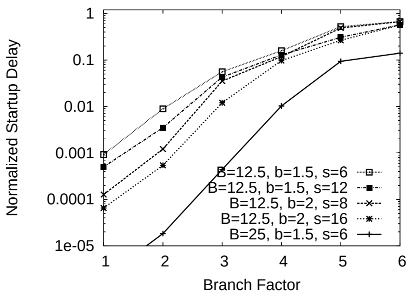


Fig. 12. Impact of Tree Width (symmetric tree of height three with a common initial media portion; all portions the same size)

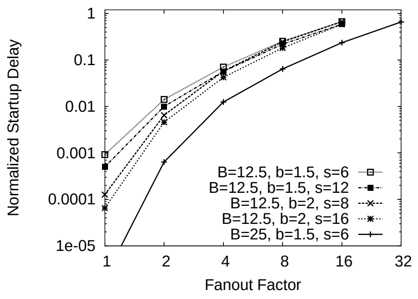


Fig. 13. Impact of Fanout Factor (media in Fig. 1(b); all portions the same size)

previously in this section, it should be noted that the solver used to solve each LP typically runs out of memory when the number of constraints in these calculations exceeds roughly $1.6 \cdot 10^4$. Therefore, the approach discussed here is limited to structures with no more constraints than the structure with $q = 64$. Of course, a stronger LP solver would extend the range of problems that can be considered. [6]

When considering these results, it should be noted that the startup delays are relatively insensitive to the number of path choices. For example, while each LP using $J = 100$ or $J = 200$ has roughly 2 or 4, respectively, times the number of constraints as for $J = 50$, the startup delays are very similar.

TABLE II
RANDOM NON-LINEAR STRUCTURE

Path Length (q)	**Startup Delay**	**Constraints**	**LPs Solved**
1	$5.2 \cdot 10^{-2}$	$6.2 \cdot 10^{2}$	$2.0 \cdot 10^{4}$
2	$4.6 \cdot 10^{-2}$	$8.0 \cdot 10^{2}$	$1.9 \cdot 10^{4}$
4	$2.4 \cdot 10^{-2}$	$1.3 \cdot 10^{3}$	$2.0 \cdot 10^{4}$
8	$8.5 \cdot 10^{-3}$	$2.4 \cdot 10^{3}$	$2.1 \cdot 10^{4}$
16	$5.4 \cdot 10^{-3}$	$4.3 \cdot 10^{3}$	$2.1 \cdot 10^{4}$
32	$2.3 \cdot 10^{-3}$	$8.3 \cdot 10^{3}$	$2.1 \cdot 10^{4}$
64	$1.4 \cdot 10^{-3}$	$1.6 \cdot 10^{4}$	$2.1 \cdot 10^{4}$

V. CONCLUSIONS

This paper has addressed the problem of devising optimized periodic broadcast delivery protocols for non-linear media. We developed an optimization model based on solution of potentially large numbers of linear programs, together with an efficient approximation algorithm for cases in which exact solutions of the optimization model are infeasible. Use of a weighted average of client startup delays as our objective function was found to enable effective control of the relative quality of service provided to clients with differing reception capacity and/or chosen playback path.

We found that optimal periodic broadcast protocols for non-linear media can differ greatly in their basic characteristics from those for linear media. Even for the context where clients have homogeneous reception capacities, optimal protocols may employ out-of-order segment retrieval, non round-robin retrieval of segments on the client reception channels, and (even for channel rates greater than the media playback rate) non-monotone segment length progressions. Nonetheless, we were able to employ pruning of the model solution space, and exploit special structure in the optimal solution for a particular class of scenarios, to find optimal solutions in many cases. Our (generally-applicable) approximation algorithm yielded solutions that were either optimal, or within 20% of optimal, in all cases considered for which comparison with the optimal was possible. Future work includes new protocol design algorithms that are applicable for even larger non-linear media structures, which build on the insights of the characteristics of the optimal solutions.

[6]For larger structures, we expect that the characteristics of the optimal solution of smaller structures can be used to design efficient heuristic algorithms. Such algorithms remain future work.

REFERENCES

[1] M. K. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley. Periodic Broadcast and Patching Services - Implementation, Measurement, and Analysis in an Internet Streaming Video Testbed. *ACM Multimedia Journal, Special Issue on Multimedia Distribution*, 9(1):78–93, July 2003.

[2] Y. Chawathe, S. McCanne, and E. Brewer. RMX: Reliable Multicast in Heterogeneous Environments. In *Proc. IEEE INFOCOM '00*, pages 795–804, Tel Aviv, Israel, March 2000.

[3] D. Eager, M. Vernon, and J. Zahorjan. Minimizing Bandwidth Requirements for On-Demand Data Delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, September/October 2001.

[4] D. Gotz. Scalable and Adaptive Streaming for Non-Linear Media. In *Proc. ACM MULTIMEDIA '06*, pages 357–366, Santa Barbara, CA, October 2006.

[5] A. Hu. Video-on-Demand Broadcasting Protocols: A Comprehensive Study. In *Proc. IEEE INFOCOM '01*, pages 508–517, Anchorage, AK, April 2001.

[6] A. Hu, I. Nikolaidis, and P. Beek. On the Design of Efficient Video-on-Demand Broadcast Schemes. In *Proc. MASCOTS '99*, pages 262–269, Maryland, MD, October 1999.

[7] K. 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, September 1997.

[8] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. OSDI '00*, pages 197–212, San Diego, CA, October 2000.

[9] L. Juhn and L. Tseng. Harmonic Broadcasting for Video-on-Demand Service. *IEEE Trans. on Broadcasting*, 43(3):268–271, September 1997.

[10] L. Lao, J.-H. Cui, and M. Gerla. TOMA: A Viable Solution for Large-Scale Multicast Service Support. In *Proc. IFIP Networking '05*, pages 906–917, Waterloo, Canada, May 2005.

[11] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable On-Demand Media Streaming with Packet Loss Recovery. *IEEE/ACM Trans. on Networking*, 11(2):195–209, April 2003.

[12] B. Qudah and N. J. Sarhan. Towards Scalable Delivery of Video Streams to Heterogeneous Receivers. In *Proc. ACM MULTIMEDIA '06*, pages 347–356, Santa Barbara, CA, October 2006.

[13] S. Rost, J. Byers, and A. Bestavros. The Cyclone Server Architecture: Streamlining Delivery of Popular Content. In *Proc. WCW '01*, pages 147–163, Boston, MA, June 2001.

[14] L. Shi, P. Sessini, A. Mahanti, Z. Li, and D. Eager. Scalable Streaming for Heterogeneous Clients. In *Proc. ACM MULTIMEDIA '06*, pages 337–346, Santa Barbara, CA, October 2006.

[15] S. Viswanathan and T. Imielinski. Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting. *Multimedia Systems*, 4(4):197–208, August 1996.

[16] Y. Zhao, D. Eager, and M. Vernon. Scalable On-Demand Streaming of Non-Linear Media. In *Proc. IEEE INFOCOM '04*, pages 1522–1533, Hong Kong, China, March 2004.

APPENDIX

This appendix provides a proof of the second property of the optimal solution structure in Section III-F. More specifically, assuming that $s_j = 2$, $r \geq 1$, and the media file has a tree structure, we want to show that there exists optimal client reception schedules for which clients of type j retrieve some initial number of segments of the root portion in round-robin fashion and the remaining segments of the root portion sequentially (over the same reception channel).

Under these constraints, it is never advantageous for clients to start downloading segments of the root portion out-of-order. Therefore, there always exists an optimal solution in which the download schedule for each client type j ensures that $t_j(k_A)-l(k_A) \leq t_j(k_B)-l(k_B)$, whenever $k_A \leq k_B$. Restricting our attention to segment size progressions and schedules satisfying this condition, we show that such solutions always can be modified into a solution with no greater startup delay for which each client schedule also satisfies the additional constraint that $t_j(k_A) \leq t_j(k_B)$. By construction, any schedule satisfying both these properties satisfies the properties outlined in Section III-F.

Our proof relies on induction on the number of segments k for which the schedules have been modified to ensure that both $t_j(k_A) \leq t_j(k_B)$ and $t_j(k_A) - l(k_A) \leq t_j(k_B) - l(k_B)$, for all $k_A \leq k_B \leq k$ and $1 \leq j \leq J$. Clearly, for $k = 1$ this property is true without any changes to the segment lengths or client schedules. Assuming the above properties are true for all segments up to and including segment k, we now claim that the sizes of segments k and $k+1$ of an optimal schedule with $t_j(k_A) - l(k_A) \leq t_j(k_B) - l(k_B)$, for all $k_A \leq k_B$ can be modified such that the properties are true for all segments up to and including segment $k+1$.

For any client type j, segments k and $k+1$ are either retrieved over the same or over different reception channels. With $t_j(k_A) \leq t_j(k_B)$, for all $k_A \leq k_B \leq k$, and $t_j(k_A) - l(k_A) \leq t_j(k_B) - l(k_B)$, for all $k_A \leq k_B \leq k+1$, any client type j which receives segments k and $k+1$ over different channels must have received all previous segments in round-robin fashion. Therefore, the schedules of all client types receiving segments k and $k+1$ over different channels are identical up to and including (at least) segment $k+1$. In contrast, the schedules of client types that receive these segments over the same channel may differ.

Consider first the client types which receive segments k and $k+1$ over different channels. The desired property holds without altering schedules or segment lengths if $t_j(k) \leq t_j(k+1)$. However, if $t_j(k) > t_j(k+1)$ we must adjust the lengths of these two segments. To avoid changing the deadlines of any other segments (either before or after segments k and $k+1$) the segment sizes of segment k and $k+1$ are adjusted such that their deadlines are exchanged. This corresponds to decreasing the size of segment k by $\Delta l = r(t_j(k) - t_j(k+1))$ and increasing the size of segment $k+1$ by the same amount. By induction, completion times are non-decreasing and therefore $l_j^{new}(k) = l_j(k) - \Delta l$ is always non-negative. Given that segment k originally was retrieved in time of its playback, both segment k and $k+1$ will be retrieved in time for playback (using the new schedule).

In addition to changing the sizes of these two segments we also change the reception channel over which any segment indexed $k+2$ or higher is retrieved, such that any segment previously scheduled on the same channel as segment k instead is scheduled on the same channel as segment $k+1$ (and vice versa). This ensures that all later completion times are preserved. With all segments retrieved by their original deadlines, the new schedule (of each client type receiving segment k and $k+1$ over different channels) always achieves at least the same startup delay as the original schedule.

Now, consider the impact the above change of the segment sizes of segment k and $k+1$ has on all other schedules, in which k and $k+1$ are scheduled (back-to-back) over the same channel. Clearly, only the slack of segment k and $k+1$ will be affected by this change. Let $\tau_{j,k} = t_j(k) - \sum_{i=1}^{k-1} l_i$ denote the minimum startup delay required to deliver segment k in time of its playback time. Using this definition, the corresponding startup delay constraints associated with segments k and $k+1$ are equal to

$$\tau_{j,k}^{new} = (t_j(k) - \Delta l/r) - \sum_{i=1}^{k-1} l_i \leq \tau_{j,k} \tag{14}$$

and

$$t_{j,k+1}^{new} = (t_j(k) + l_{k+1}/r) - \sum_{i=1}^{k-1} l_i + (l_k - \Delta l) \leq \tau_{j,k}. \tag{15}$$

The last inequality uses the fact that $\Delta l \leq (l_k - l_{k+1}) \leq (l_k - l_{k+1}/r)$, when $r \geq 1$. With neither of the segments requiring an increase in the startup delay, the above properties are true for $k+1$. This completes the proof.