Scalable On-Demand Media Streaming for Heterogeneous Clients

PHILLIPA GILL and LIQI SHI
University of Calgary
ANIRBAN MAHANTI
Indian Institute of Technology
ZONGPENG LI
University of Calgary
and
DEREK L. EAGER
University of Saskatchewan

Periodic broadcast protocols enable efficient streaming of highly popular media files to large numbers of concurrent clients. Most previous periodic broadcast protocols, however, assume that all clients can receive at the same rate, and also assume that reception bandwidth is not time-varying. In this article, we first develop a new periodic broadcast protocol, Optimized Heterogeneous Periodic Broadcast (OHPB), that can be optimized for a given population of clients with heterogeneous reception bandwidths and quality-of-service requirements. The OHPB protocol utilizes an optimized segment size progression determined by solving a linear optimization model that takes as input the client population characteristics and an objective function such as mean client startup delay. We then develop a generalization of the OHPB linear optimization model that allows optimal server bandwidth allocation among multiple concurrent OHPB broadcasts, wherein each media file and its clients may have different characteristics. Finally, we propose complementary client protocols employing work-ahead buffering of data during playback, so as to enable more uniform playback quality when the reception bandwidth is time-varying.

Categories and Subject Descriptors: C.2.2 [**Computer Communications Networks**]: Network Protocols; C.4 [**Computer Systems Organization**]: Performance of Systems; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms: Design, Performance

Additional Key Words and Phrases: Scalable streaming, periodic broadcasts, linear programming, quality-of-service

ACM Reference Format:

Gill, P., Shi, L., Mahanti, A., Li, Z., and Eager, D. L. 2008. Scalable on-demand media streaming for heterogeneous clients. ACM Trans. Multimedia Comput. Commun. Appl. 5, 1, Article 8 (October 2008), 24 pages. DOI = 10.1145/1404880.1404888 http://doi.acm.org/10.1145/ 1404880.1404888

A preliminary version of this article was presented at Multimedia 2006.

Authors' addresses: P. Gill, Z. Li, Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada; email: {psessini, zongpeng}@cpsc.ucalgary.ca. L. Shi, Department of Electrical & Computer Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada; email: lishi@ucalgary.ca; A. Mahanti, Department of Computer Science & Engineering, Indian Institute of Technology, Delhi, India 110 016; email: mahanti@cse.iitd.ac.in; D. L. Eager, Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5C9, Canada; email: eager@cs.usask.ca.

1. INTRODUCTION

Media-on-Demand systems that utilize *periodic broadcast* protocols can concurrently serve large numbers of clients with low start-up delay using a fixed amount of server resources by utilizing the one-to-many dissemination capability of networks (e.g., application or native multicast in the Internet). Previously proposed periodic broadcast protocols include, among others, Pyramid Broadcasting [Viswanathan and Imielinski 1996], Harmonic Broadcasting [Juhn and Tseng 1997], Skyscraper Broadcasting [Hua and Sheu 1997], Dynamic Skyscraper [Eager and Vernon 1998], and more recently, the Optimized Periodic Broadcast protocol family [Mahanti et al. 2003]. Although the aforementioned protocols achieve differing tradeoffs, for example between start-up delay and server bandwidth, they all operate within a common framework. More specifically, these protocols partition a media file into some number of segments and cyclically multicast these segments using a fixed number of server channels (e.g., multicast groups), according to some predetermined schedule. Clients requesting a media file are given a start-up delay to begin playback and a schedule for tuning into the server channels to receive segments such that all data will be received by the time it is required for playback. Typically, clients receive multiple segments concurrently at an aggregate rate that exceeds the media playback rate and buffer data that is received ahead of playback time.

All previous periodic broadcast protocols, with the exception of the recently proposed Heterogeneous Receiver-Oriented Broadcasting (HeRO) [Hua et al. 2003] and BroadCatch [Tantaoui et al. 2004] protocols that we discuss later in the article, assume that all clients can receive media streams at the same non-time-varying rate. This homogeneous client bandwidth assumption is unlikely to hold for delivery in the Internet. For example, many service providers offer users a choice of different levels of broadband Internet connectivity, with each level providing different last hop outbound/inbound bandwidth.

One strategy for accommodating heterogeneous client reception rates is to use either a *simulcast* [Cheung et al. 1996; Kim and Ammar 2001] or a *layered media encoding* [Li et al. 1999; McCanne et al. 1996]. With the simulcast approach different versions of the media file are encoded at different bit rates, each of which can be independently delivered using periodic broadcast. Alternatively, with a layered media encoding, a periodic broadcast scheme may be used to deliver each layer. In either case, a priori knowledge of the average reception bandwidth can help clients determine which layers (or version) to receive. Using layering or simulcasting with a periodic broadcast scheme tailored for a single bandwidth provides only limited support for client heterogeneity. Specifically, such solutions do *not* allow efficient tradeoffs between start-up delay and media quality. For example, a client with slightly lower (average) bandwidth than that required to receive media of some desired quality (i.e., layers or version) requires a relatively large increase in the start-up delay to guarantee continuous playback of the higher quality content.

Furthermore, in the “best effort” Internet, it is recommended that UDP-based media streaming applications share bandwidth fairly with TCP-based applications by using an appropriate rate control protocol [Floyd and Fall 1999]. In streaming media applications, this rate control is typically implemented by modifying the amount of data sent, thus implying a change in playback quality, in contrast to traditional TCP applications in which the same amount of data is sent, but at a slower rate. Scalable streaming systems might use multirate congestion control algorithms [Legout and Biersack 2000; Li and Ammar 1996; Li et al. 1999; Luby et al. 2002; Mahanti et al. 2005; Vicisano et al. 1998] which result in time-varying reception bandwidths. For the case of layered media files (or replicated streams), however, it is not apparent how layers (or versions) should be added/dropped (switched), so that overall playback quality is both relatively uniform and high.

In this paper, we address the challenge of devising efficient periodic broadcast systems for environments with both heterogeneity in the average client reception bandwidth, and heterogeneity due to time-varying reception bandwidth. We make three contributions. First, we develop a new periodic broadcast protocol, Optimized Heterogeneous Periodic Broadcast (OHPB), that can be *optimized* for a given population of clients with heterogeneous reception bandwidths and quality-of-service requirements. Second, we develop a generalization of OHPB, OHPB-Concurrent (OHPB-C), that facilitates optimal server bandwidth allocation among multiple media files. Third, for delivery of layered media files, we develop complementary *quality adaptation* [Rejaie et al. 1999a] mechanisms and policies that allow each client to independently determine how to allocate time-varying reception bandwidth among layers at any instant of time so as to achieve playback quality that is as uniform and as high as possible.

The crux of the new OHPB protocol is the technique by which the segment size progression is computed. Different segment size progressions require different start-up delays at a client with a certain reception bandwidth. For systems with homogeneous clients, this delay optimization problem has been effectively addressed [Mahanti et al. 2003; Mahanti 2004]. However, designing an optimal progression for a population of clients with heterogeneous reception bandwidths is a new problem. The challenge lies essentially in making good tradeoffs among the start-up delays experienced by the different clients. In this article, we model such an optimization problem as a Linear Program (LP) and propose an efficient subgradient algorithm for solving this linear program.

Similar to many previous periodic broadcast protocols, OHPB assumes that a single media file is to be broadcast. A media on-demand server, however, may need to broadcast multiple media files, wherein each media file is potentially associated with a different population of clients and reception bandwidth characteristics. Thus, we develop a generalization of OHPB, OHPB-C, that allows determination of the optimal allocation of server bandwidth among multiple concurrent broadcasts of popular media files. Our solution is in the form of a Mixed Integer Linear Program (MILP) that takes as input the available server bandwidth, the characteristics of clients accessing the media files, and an optimization criteria such as optimal start-up delay of sessions with min-max fairness. In general, linear programs with integer variables are computationally expensive to solve. For the proposed OHPB-C MILP, we develop an efficient solution algorithm that exploits the structure of this MILP.

For delivery of layered media files using OHPB systems, we develop efficient mechanisms and policies to handle time-varying client reception bandwidth. Note that periodic broadcast systems multicast segment transmissions, and therefore, altering server-side transmission rates due to conditions at any one particular client is not feasible. The only option is to use client-side policies for work-ahead (i.e., buffering data prior to when it is needed for playback) that are facilitated by listening on a channel earlier than when required by the protocol. Such work-ahead is difficult to implement for most periodic broadcast systems owing to the cyclic nature of segment transmissions. For example, if a client obtains only a portion of a media segment during work-ahead and then drops the channel due to a bandwidth change, this work-ahead may not prove useful because when the receiver again begins to listen to the channel, the portion of the segment being currently transmitted may substantially overlap with the buffered portion. Our contribution in this context is the design of an efficient quality adaptation mechanism for layered media files delivered using OHPB systems. A key element of our solution is to treat the transmission of each segment as a *digital fountain* [Byers et al. 1998; Rizzo and Vicisano 1997] to avoid the aforementioned problems with cyclic segment transmissions.

This article is structured as follows. Section 2 presents background on scalable multicast/broadcast on-demand streaming systems, and on quality adaptation for streaming media. Section 3 describes the OHPB protocol for heterogeneous clients, outlining the optimization technique used to obtain the segment size progression and the advantages of this scheme with respect to prior proposals. Section 4 presents OHPB extensions for multiple concurrent broadcasts. Our quality adaptation proposal is presented in Section 5. Conclusions are presented in Section 6.

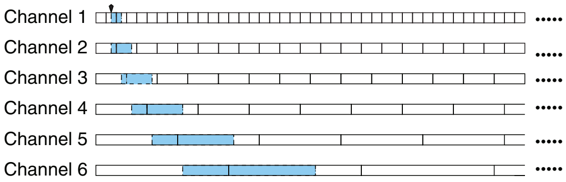


Fig. 1. Optimized PB ($K = 6, r = 1, s = 2$).

2. BACKGROUND AND RELATED WORK

Content delivery systems that concurrently serve large numbers of clients may use proxy servers that replicate popular content at the network edge, use application or network layer multicast, or use a combination of these approaches [Almeida et al. 2002; Eager et al. 2000, 1999; Sen et al. 1999; Wang et al. 2002]. For on-demand streaming of large, popular media files, use of scalable multicast streaming protocols is expected owing to the impressive server and network bandwidth savings achievable using this approach [Eager et al. 2001; Hu 2001; Zhao et al. 2002].

Among the scalable multicast streaming protocols, the *periodic broadcast* [Gao et al. 1998; Hua and Sheu 1997; Juhn and Tseng 1997; Mahanti et al. 2003; Viswanathan and Imielinski 1996] protocols and the *stream merging* [Bar-Noy et al. 2002; Eager et al. 1999, 2000, 2001; Qudah and Sarhan 2006] protocols are the most effective. The server bandwidth requirements of stream merging protocols grow as a function of client request rate. This work focuses on periodic broadcast protocols because their server bandwidth requirement is independent of the client request rate, and therefore, they are better suited for streaming the most popular files.

Section 2.1 reviews the Optimized Periodic Broadcast (Optimized PB) protocol [Mahanti et al. 2003; Mahanti 2004]. Our work uses this protocol as a building block because: 1) unlike other protocols, it can support any client reception bandwidth, even reception bandwidths that are less than twice the playback rate; and 2) it can be extended to support efficient packet loss recovery. Sections 2.2 and 2.3 review two recent proposals for supporting heterogeneous client bandwidths using periodic broadcast, the BroadCatch [Tantaoui et al. 2004] and the Heterogeneous Receiver-Oriented Broadcasting (HeRO) [Hua et al. 2003] protocols, respectively. Section 2.4 reviews work on quality adaptation for unicast streaming and discusses the difficulties that arise when applying similar approaches for multicast streaming.

2.1 Optimized Periodic Broadcast

In the Optimized PB protocol, a media file is divided into K segments, each of which is repeatedly transmitted on its own channel at rate r times the media playback rate. Each client that requests the file immediately begins listening to the first channel, and commences playback once the first segment is completely received. Concurrently, the client listens to the channels delivering the next $s-1$ segments. Once a segment has been completely received on a channel i, the client immediately begins listening to channel $i+s$ (for $i+s \leq K$). The progression of segment sizes (dependent on the transmission rate r of each segment and the number of channels s that each client listens to concurrently) is such that data is received just-in-time for playout. Figure 1 shows an example Optimized PB broadcast where each client listens simultaneously to a total of two channels; periods during which an example client that requests the file at the time indicated by the arrow listens to each channel are denoted by the shaded regions.

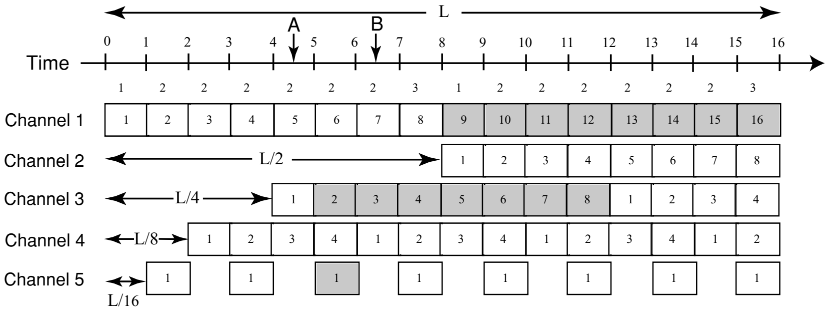


Fig. 2. BroadCatch ($K = 5, r = 1$).

The Optimized PB protocol uses an optimal (in the sense of minimizing client start-up delay for a given number of server channels K) segment size progression for any given value $r > 0$ and integer $s > 1$. Thus, as with other previous periodic broadcast protocols, the segment size progression is designed (explicitly or implicitly) for some particular achievable client bandwidth. Substantial client heterogeneity is assumed to be handled using layered media and a scheme whereby clients match their layer subscription to the available bandwidth on their path from the server, so that all clients receiving a layer are able to concurrently listen to the required number of channels.

2.2 BroadCatch

The BroadCatch protocol divides a media file into 2^{K-1} equal-sized segments, with a contiguous group of these segments being transmitted at playback rate (i.e., $r = 1$) on K separate channels [Tantaoui et al. 2004]. The first two channels cyclically broadcast the entire media file, with the beginning of the transmission on the second channel offset by $L/2$ time units (where L is the media playback duration) with respect to the beginning of the transmission on the first channel. On any channel i, $3 \leq i < K$, the server cyclically broadcasts the first 2^{K-i+1} segments with the first broadcast on this channel offset with respect to the first broadcast on channel 1 by $\frac{L}{2^{i-1}}$ time units. Channel K broadcasts only the first segment, with the broadcasts coinciding with every alternate segment broadcast on channel 1.

Figure 2 presents an example BroadCatch broadcast with $K = 5$ channels and illustrates the sequence of segments received by a client A that arrives between time 4 and 5; this client has bandwidth to concurrently listen on two channels. For each segment 1 broadcast on any of the channels, there is an associated minimum client bandwidth that would be required for a client to begin playback immediately upon beginning reception of that broadcast. In Figure 2, these minimum client bandwidths, as measured by the number of channels a client must concurrently listen to, are given just above the row showing the transmission schedule of Channel 1. In our example, client B with bandwidth equal to the media playback rate that arrives between time 6 and 7 waits until time 8 to begin playback.

The BroadCatch scheme accommodates clients with bandwidths in the range 1 to $(K - 2)$ times the media playback rate. Increasing K reduces the segment sizes, and thus reduces the start-up delay for the clients with the highest reception bandwidths. However, in most cases, only clients with the capability of receiving at a high rate are able to take advantage of the improved start-up delay.

2.3 Heterogeneous Receiver-Oriented Broadcasting

The HeRO protocol is another technique for clients with heterogeneous bandwidths [Hua et al. 2003]. This protocol partitions a media file into K segments with the relative segment sizes $1, 2, 2^2, \cdots, 2^{K-1}$. Each segment is broadcast on a separate channel at the playback rate. To better accommodate heterogeneous client bandwidths, each of the last $K_s \geq 0$ segments are broadcast on two channels rather than one, with the broadcasts of the segment on the second channel staggered with respect to those on the first channel by half the segment length. The increased broadcast frequency of the large segments enables reduced start-up delay for clients with reception rate less than K. As illustrated by the numerical results presented in Section 3.3, HeRO and BroadCatch have similar performance.

2.4 Quality Adaptation

The most closely related previous work on quality adaptation considered unicast streaming of layered media files [Rejaie et al. 1999a]. In that work, a TCP-friendly rate adaptation policy [Rejaie et al. 1999b] is utilized to determine the transmission rate of the server. During high rate periods, work-ahead is achieved by transmitting the data for certain layers (determined by the work-ahead algorithm) at a rate higher than its consumption rate. This work-ahead allows the server to reduce the sending rate on layers that have sufficient work-ahead during low rate periods. In this manner, work-ahead is used to smooth short-term variations in the available bandwidth. In response to long-term changes in the available bandwidth, the server adds or drops media layers.

With periodic broadcast systems that employ multirate congestion control (e.g., [Legout and Biersack 2000; Li and Ammar 1996; Li et al. 1999; Luby et al. 2002; Mahanti et al. 2005; Vicisano et al. 1998; Byers and Kwon 2001; Turletti et al. 1997; Widmer et al. 2001]), it is not feasible to change the transmission rate on a channel to accommodate the needs of an individual client because it may adversely affect other clients listening to the channel. Thus, rate adjustments must be made by the client rather than by the server, by adding or dropping multicast channels. The challenge in the context of periodic broadcasting is to devise protocols in which clients can make rate adjustments in this manner, and yet exploit work-ahead to smooth short-term fluctuations in the client reception bandwidth.

With broadcast protocols such as Harmonic Broadcasting [Juhn and Tseng 1997] and its variants [Hu et al. 1999; Paris et al. 1998], in which clients listen concurrently to all of the server channels that are transmitting segments of the requested file, clients cannot work-ahead unless the server somehow transmits additional streams for this purpose. With Optimized PB, BroadCatch, HeRO, or any other protocols wherein clients listen to a subset of channels, a client that temporarily has extra bandwidth might work-ahead by listening to extra channels. However, if the client has to drop one of these extra channels due to a temporary decrease in reception bandwidth, then when it later returns to this channel, it may have to wait a considerable length of time before the remaining data that it needs for that segment is transmitted again.

3. THE OHPB PROTOCOL

Assume that there are N distinct types of clients, where the clients of type j have reception bandwidth b_j. The problem addressed in the design of the Optimized Heterogeneous Periodic Broadcast (OHPB) protocol is that of devising a segment size progression that yields the best possible performance, according to some given objective function, for a given population of clients. Section 3.1 develops the new OHPB protocol and outlines the OHPB linear program (LP). Section 3.2 discusses an example OHPB broadcast. Numerical results illustrating OHPB performance, and comparisons with HeRO and BroadCatch, are presented in Section 3.3. Possible tradeoffs between media quality and start-up delay for layered media files are discussed in Section 3.4. We conclude this section with a qualitative discussion of OHPB's salient features. Our discussion in this section is applicable for the delivery of a single monolithic media file or a single layer of a layer-encoded media file; delivery of multiple files is discussed in Section 4. We also note that our discussion here assumes that the rate at which each client can receive media data does not change substantially during its session; this assumption is relaxed in Section 5. For ease of reference, Table I summarizes the notation used in the OHPB LP.

Table I. Notation for the OHPB Protocol

Symbol	Definition
K	Total number of segments (server channels)
l_i	Playback duration of the ith segment
$t_j(k)$	Time required by a type j client to complete download of the first k segments of the media object
L	Total media playback duration
r	Segment transmission rate (in units of media playback rate)
B	Server bandwidth (in units of media playback rate; $B = K \times r$)
N	Number of client types
b_j	Bandwidth of type j clients; $b_j \geq r$
s_j	Number of channels a type j client can concurrently listen on; $s_j = \min(\lfloor b_j/r \rfloor, K)$
τ_j	Deterministic start-up delay of type j clients
w_j	Weight used for clients of type j

3.1 Design of OHPB

The OHPB protocol adopts the following general framework, similar to a number of other periodic broadcast protocols:

—The media file, of playback duration L, is partitioned into K segments; each segment i, $1 \leq i \leq K$, is of duration l_i where $\sum_i l_i = L$. It is assumed here that the media file is constant bit rate (although generalizations are possible).

—Each segment i is repeatedly multicast on server channel i at r times the media playback rate. Thus, the total required server bandwidth B is equal to $K \times r$, in units of the media playback rate.

—On receiving a request for the media file, the server provides the requesting client with a start-up delay and a schedule for tuning into the channels. The start-up delay τ_j for clients of type j is *deterministic*.

—Each segment is completely downloaded prior to commencing its playback. This approach makes OHPB amenable to the packet loss recovery approach used in Optimized PB [Mahanti et al. 2003], and furthermore, allows design of flexible quality adaptation techniques. A consequence of this approach is that the minimum possible start-up delay is the time required to download the first segment.

Within this framework, we consider the problem of optimizing the segment size progression for a given population of clients. For type j clients, let $t_j(k)$ denote the time required to complete downloading the first k segments. Because a type j client can concurrently listen to s_j channels, where $s_j = \min(\lfloor b_j/r \rfloor, K)$, we have the following equations:

$$t_j(k) = \frac{l_k}{r}, 1 \leq k \leq s_j, s_j \geq 1, \tag{1}$$

$$t_j(k) = t_j(k - s_j) + \frac{l_k}{r}, s_j < k \leq K, s_j \geq 1. \tag{2}$$

OHPB requires that each segment k, $1 < k \leq K$, be entirely downloaded by the time segment $k - 1$ finishes playback. Furthermore, the first segment must be available in the client's buffer following the start-up delay τ_j. Therefore, the following relation must be satisfied:

$$t_j(k) \leq \tau_j + \sum_{i=1}^{k-1} l_i, 1 \leq k \leq K. \tag{3}$$

Many choices of segment sizes satisfy the above constraints. Our problem is to choose segment sizes that are optimal for a chosen objective function. Initially, we consider an objective function in which each client type is assigned a weight w_j. This weight could reflect, for example, the fraction of the total requests for the media file that are generated by type j clients. The objective function is then chosen as the weighted average of the start-up delays for the N types of client, yielding the following OHPB linear program (LP):
Minimize

$$M1 = \sum_j w_j \tau_j \tag{4}$$

Subject to:

$$\sum_i l_i = L \tag{5}$$

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

$$t_j(k) = \frac{l_k}{r} \quad \forall j, 1 \le k \le s_j \tag{7}$$

$$t_j(k) = t_j(k - s_j) + \frac{l_k}{r}, \quad \forall j, s_j < k \le K \tag{8}$$

$$l_i, \tau_j, t_j(k) \ge 0, \quad \forall i, j, k. \tag{9}$$

The inputs to the OHPB LP are K, r, N, s_j's, w_j's, and L. The solution to the LP gives the sequence of segment sizes (l_i's), and the start-up delay for each client type (τ_j). The number of variables and the number of constraints of the above LP are both $O(NK)$. We have developed a subgradient algorithm tailored for the OHPB LP. It is a stand-alone algorithm that can be executed without using a third-party LP solver, and consists of only combinatorial steps. This algorithm, presented in the appendix of the article, exploits the specific underlying structure of the OHPB LP and achieves much higher scalability than general simplex or interior-point algorithms.

In general, the choice of the objective function depends on the specific quality-of-service goals of the server provider. Function $M1$ is one among the many possible objective functions. For equal weight assignment to each client type, this function attempts to reduce start-up delays for low bandwidth clients at the cost of increasing start-up delays for high bandwidth clients. Alternatively, if clients with higher bandwidth make requests more often, a broadcast scheme to favor these clients may be designed by assigning these clients a higher weight in the optimization function. Many other variations are possible. As one more example, clients may be divided into types, and weights assigned, in part according to the level of delivery service purchased, rather than just the client bandwidth.

We illustrate the power of the proposed technique by considering another objective function. This second objective function considers the factor increase in start-up delay that each type of client experiences when OHPB is used, compared to the Optimized PB protocol tailored for that client type. Specifically, let τ_j^{opb} denote the start-up delay using an Optimized PB protocol tailored to achieve the minimum start-up delay for client type j, given server bandwidth $B = K \times r$. The second objective function is then given as follows:
Minimize

$$M2 = \sum_j w_j \frac{\tau_j}{\tau_j^{opb}}. \tag{10}$$

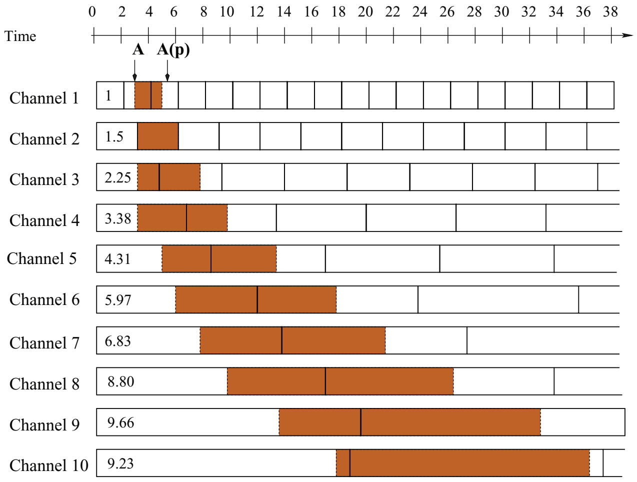


Fig. 3. OHPB ($B = 5$, $r = 0.5$, client bandwidths [1, 1.5, 2, 2.5, 3, 3, 5, 4] with equal weight assignment, model 1).

Note that for fixed s_j, r, and K, τ_j^{opb} is a constant, and thus, $M2$ is a linear function. In the following sections, the OHPB linear programs obtained using functions $M1$ and $M2$ are referred to as models 1 and 2, respectively.

3.2 An Example OHPB Broadcast

Figure 3 shows an example OHPB broadcast obtained by solving the OHPB LP (model 1) for $B = 5$, client bandwidths [1, 1.5, 2, 2.5, 3, 3.5, 4] with each client type assigned an equal weight, and segment transmission rate $r = 0.5$; B, r, and client bandwidths are in units of the media playback rate. As with Optimized PB, each segment is cyclically broadcast on a separate channel. In the illustration, the relative lengths of segments are marked on the first segment broadcast of each channel. The figure also shows the sequence of segments received by a client A that has the ability to simultaneously receive on 4 channels (i.e., $b = 2$). Similar to Optimized PB and as shown in the example, a client with bandwidth b_j upon arrival immediately begins downloading segments on s_j channels. In this example OHPB broadcast, clients that have bandwidths greater than or equal to 2.5 require start-up delays that equal the time to download segment 1 (which is the minimum possible start-up delay for this example broadcast). Clients that have bandwidth less than 2.5 require longer start-up delays to enable uninterrupted playback. For example, Client A in our example begins playback at time marked $A(p)$, which is after the download of segment 1 but before completion of download of segment 2. More specifically, the start-up delay for this client is 0.047 times the media playback duration. The following sections present numerical results of OHPB performance for a variety of scenarios, as well as qualitative and quantitative comparisons with previously proposed protocols, namely BroadCatch and HeRO.

3.3 Numerical Results

Figure 4 shows the required start-up delay for specific client bandwidths as a function of server bandwidth. In this and subsequent results, unless stated otherwise, server bandwidth is expressed in units of the media playback rate, and start-up delay is expressed as a fraction of the media file playback duration. The results in the figure are obtained by solving the OHPB LP (model 2) with client bandwidths [1, 1.5, 2, 2.5, 3, 3.5, 4], with each client type assigned an equal weight, and segment transmission rate $r = 0.25$. The results indicate that *linear* increases in server bandwidth yield *exponential* decreases in

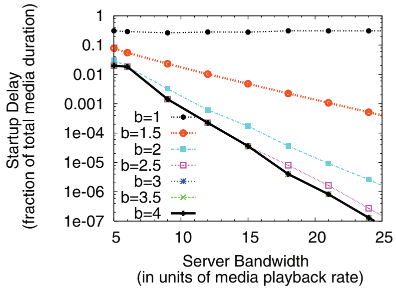


Fig. 4. Scalability of OHPB (model 2, $r = 0.25$).

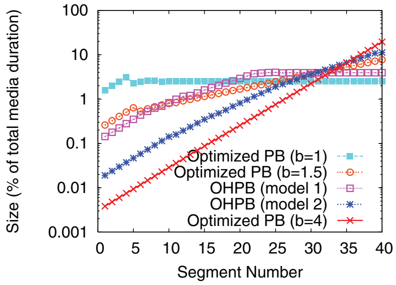


Fig. 5. Segment size progressions ($B = 10$, $r = 0.25$).

start-up delay for clients with data rates greater than the media playback rate (i.e., $b > 1$); this property has been observed for other periodic broadcast protocols [Hu 2001]. For clients with bandwidth equal to the media playback rate, increasing server bandwidth has negligible impact on start-up delay because the client must buffer a large fraction of the media file before playback can begin. Qualitatively similar results are obtained for OHPB model 1. With OHPB model 1, however, low bandwidth clients obtain preferential treatment over high bandwidth clients because the higher start-up delays experienced by the low bandwidth clients dominate the objective function for model 1.

Examining the OHPB segment size progression provides further insights. For $B = 10$ and $r = 0.25$, Figure 5 shows the segment size progressions for OHPB models 1 and 2 with client bandwidths [1, 1.5, 2, 2.5, 3, 3.5, 4] and equal weight assignment for each client type. The figure also shows the Optimized PB segment size progressions for $b = 1$, $b = 1.5$, and $b = 4$. Note that OHPB model 2 yields exponentially increasing segment sizes most similar to the exponentially increasing segment sizes of Optimized PB for $b = 4$. Compared to Optimized PB, however, OHPB exhibits slower growth of segment sizes, as would be expected when trying to accommodate heterogeneous client bandwidths. In general, accommodating clients with low reception bandwidths also requires larger initial segments. From the figure, we observe that with OHPB model 1 segment sizes grow significantly slower than with OHPB model 2, because with OHPB model 1 start-up delays are lowered for low bandwidth clients at the cost of increased delays for the high bandwidth clients. In fact, the initial segment sizes with OHPB model 1 are similar to those with Optimized PB ($b = 1.5$), while the later segment sizes are similar to those of Optimized PB ($b = 1$).

Figure 6 explores the client start-up delays for different weight assignments. We report results obtained by solving the OHPB LP (model 2) for $B = 10$, $r = 0.25$, and client bandwidths [1, 1.5, 2, 2.5, 3, 3.5, 4]. In the figure, four example weight assignments for OHPB model 2 are considered, namely: 1)

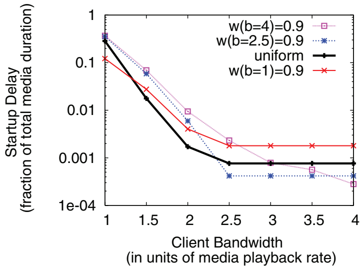
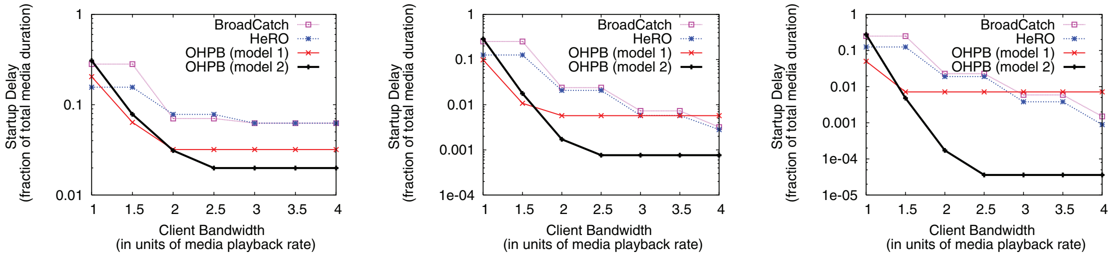


Fig. 6. Adaptivity of OHPB (model 2, $B = 10$, $r = 0.25$).



(a) $B = 5$ (b) $B = 10$ (c) $B = 15$

Fig. 7. Comparing OHPB, BroadCatch, and HeRO ($r = 0.25$).

the client type with bandwidth $b = 1$ is assigned weight 9/10, with the remaining types assigned equal weight of 1/60; 2) similar to (1) but with the client type with bandwidth $b = 2.5$ the one that is assigned weight 9/10; 3) similar to (1) but with the client type with bandwidth $b = 4$ the one that is assigned weight 9/10; and 4) all client types assigned equal weight of 1/7. The results show that assigning higher weight to a client type can significantly lower the start-up delay for that type of client, compared to that achieved with the equal weight assignment. For example, assigning a weight of 0.9 to the client type with $b = 2.5$ results in a factor of 4 reduction in start-up delay for these clients compared to when all client types have equal weight assignments. This decrease in start-up delay, however, comes at the cost of increased start-up delays for the lower bandwidth clients.

We compare OHPB with HeRO and BroadCatch for the same example set of client bandwidths ([1, 1.5, 2, 2.5, 3, 3.5, 4]) used for Figures 4–6. Figure 7 shows the start-up delay experienced by clients of each type, for three server bandwidth values. The results for OHPB are for equal weightings of the client types. The start-up delay results for BroadCatch and HeRO are averages over all distinct time slots in the respective transmission schedule, of the start-up delay for a client arriving in that time slot. Note that OHPB provides lower start-up delay than both HeRO and BroadCatch for most client bandwidths. OHPB model 2, in particular, provides substantially lower start-up delay than BroadCatch or HeRO for all client bandwidths except $b = 1$. Note also that with increasing server bandwidth but fixed client bandwidths, the relative performance of both BroadCatch and HeRO worsen since these protocols are designed for a client bandwidth range that varies with the server bandwidth. For example, when $B = 10$, BroadCatch assumes client bandwidths ranging from 1 to 8. Finally, note that, unlike OHPB, neither BroadCatch nor HeRO can take advantage of fractional units of client bandwidth, and so, for example, clients with $b = 2$ and $b = 2.5$ experience identical start-up delay.

3.4 Start-up Delay vs. Quality Tradeoffs

Consider a system where each layer of a media file is delivered using a separate instance of the OHPB protocol. It is advantageous in such a system for each instance of the protocol to use the same parameters r (measured in units of the bit rate of the respective layer) and K, and the same segment size progression. For a concrete example, we consider a system of this type that is broadcasting a 30 minute video with 10 equal bit rate layers, and, for each layer, use of OHPB (model 2) with client bandwidths and other parameters identical to those used for Figure 7(a).

Now consider a client with total available reception bandwidth $b = 10$. Such a client has per-layer bandwidth $b = 1, 1.5, 2.0$, or 2.5 if 10, 6, 5, or 4 layers are received, respectively. Thus, from Figure 6, it can be seen that the client has a choice of receiving all 10 layers with a start-up delay of 9 minutes, 6 layers with a start-up delay of 2 minutes, 5 layers with a start-up delay of 56 seconds, or 4 layers with a start-up delay of 36 seconds. There is no advantage to receiving less than 4 layers, as it does not reduce the start-up delay below 36 seconds. Had the same file been broadcast using the Optimized PB protocol with $b = 2.5$, for example, the choices available to the client would be to receive 10 layers with a start-up delay of 18 minutes, 6 layers with a start-up delay of 8 minutes, 5 layers with a start-up delay of 3 minutes, or 4 layers with a start-up delay of 27 seconds. Had the file been broadcast using the BroadCatch protocol (using $B = 5$ per layer as in Figure 7(a)), the client could receive 10 layers with a start-up delay of 8 minutes, 5 layers with a start-up delay of 2 minutes, or 3 layers with a start-up delay of 1.9 minutes. Because BroadCatch cannot take advantage of fractional bandwidth, the available quality/delay options are somewhat limited. Clearly, OHPB offers a much more flexible tradeoff between media quality and start-up delay.

3.5 Discussion

We close this section with a discussion of OHPB's salient features, and by qualitatively comparing OHPB with BroadCatch and HeRO. Note that the OHPB LP allows the media segment sizes to be tailored to satisfy any desired linear objective function (such as minimizing the mean start-up delay of clients). Neither BroadCatch or HeRO include such optimization criteria. The OHPB LP allows the segment size progression to be designed for any range of client bandwidths. In particular, the range of client bandwidths that can be supported by the protocol is independent of the server bandwidth, whereas in BroadCatch and HeRO, the supported client bandwidth range is directly related to the server bandwidth. Furthermore, since OHPB requires a segment to be completely downloaded before playback, the Reliable Periodic Broadcast [Mahanti et al. 2003; Mahanti 2004] approach of delivering each segment as a *digital fountain* [Byers et al. 1998] can be used to provide support for packet loss recovery. With protocols such as HeRO and BroadCatch that may play a segment while it is being received, this approach to packet loss recovery is not applicable. Finally, as discussed in Section 5, OHPB can be extended to support quality adaptation when the client reception bandwidth is time-varying.

4. MULTIPLE CONCURRENT BROADCASTS

The discussion of OHPB in the previous section assumed broadcast of a single media file. The OHPB model can be generalized to handle the case where multiple media files are broadcast by the same server concurrently. A fundamental question here is how to effectively and fairly share the available server bandwidth (i.e., the total number of server channels) among different media files. Section 4.1 develops the OHPB-Concurrent (OHPB-C) protocol and presents the mathematical model that governs the segment size progression of the concurrent broadcasts. Section 4.2 discusses how the mathematical model may be solved. This is followed by numerical results in Section 4.3. Notation used in this section is summarized in Table II.

Table II. Notation for Concurrent OHPB Broadcasts

Symbol	Definition
M	Total number of media files
K	Total number of segments (server channels)
K_m	Total number of segments in media file m
l_{mi}	Playback duration of the ith segment of media object m
$t_{mj}(k)$	Time required by a type j client to complete download of the first k segments of media file m
L_m	Total media playback duration of media file m
r	Segment transmission rate (in units of media playback rate)
B	Server bandwidth (in units of media playback rate; $B = K \times r$)
N	Number of client types
b_j	Bandwidth of type j clients; $b_j \geq r$
s_j	Number of channels a type j client can concurrently listen on; $s_j = \min(\lfloor \frac{b_j}{r} \rfloor, K)$
τ_{mj}	For media file m, the deterministic start-up delay for client type j
w_{mj}	For media file m, the weight used for clients of type j

4.1 Design of OHPB-C

In this section, the OHPB LP is generalized to develop a mathematical model for the optimized delivery of multiple media files. This generalization leads to a Mixed Integer Linear Program (MILP) which we refer to as OHPB-C. The OHPB-C MILP allows us consideration of a variety of optimization criteria as illustrated below.

Suppose that a service provider is interested in fairness among the concurrent broadcasts. To emphasize fairness, the objective function M3 that minimizes the maximum over the media files of the weighted average start-up delay for that file can be used to obtain the following MILP:
Minimize

$$M3 = \max_m \sum_j w_{mj} \tau_{mj} \tag{11}$$

subject to:

$$\sum_{i=1}^{K_m} l_{mi} = L_m \quad \forall m \tag{12}$$

$$t_{mj}(k) \leq \tau_{mj} + \sum_{i=1}^{k-1} l_{mi} \quad \forall m, j, k \tag{13}$$

$$t_{mj}(k) = \frac{l_{mk}}{r} \quad \forall m, j, 1 \leq k \leq s_j \tag{14}$$

$$t_{mj}(k) = t_{mj}(k - s_j) + \frac{l_{mk}}{r} \quad 1 \leq s_j < k \leq K_m \tag{15}$$

$$\sum_m K_m = K \tag{16}$$

$$l_{mi}, \tau_{mj}, t_{mj}(k) \geq 0 \quad \forall m, i, j, k; K_m \in \mathbf{Z}^+. \tag{17}$$

As another example, with the following objective function M4 the total of the weighted average start-up delays is minimized:

```
1 Choose initial channel allocation ($e.g.$, equal value of $K_m$ for every $m$);
2 $\forall m$ Solve OHPB LP for session $m$ with $K_m$ channels;
3 $m_{max} \leftarrow \text{argmax}_m \sum_j w_{mj}\tau_{mj}$; $m_{min} \leftarrow \text{argmin}_m \sum_j w_{mj}\tau_{mj}$;
  $\tau_{max} \leftarrow \sum_j w_{m_{max}j}\tau_{m_{max}j}$;
4 $K_{m_{max}} \leftarrow K_{m_{max}} + 1$, $K_{m_{min}} \leftarrow K_{m_{min}} - 1$;
5 Solve OHPB LP for $m_{max}$ and $m_{min}$ again with updated number of channels;
6 if ($\tau_{max} < \max(\sum_j w_{m_{max}j}\tau_{m_{max}j}, \sum_j w_{m_{min}j}\tau_{m_{min}j})$) then
7     output current allocation and terminate;
8 end
9 Goto 3;
```

Fig. 8. Guided local search for OHPB-C.

Minimize

$$M4 = \sum_m \sum_j w_{mj}\tau_{mj}. \tag{18}$$

Essentially, the OHPB-C MILP includes an instance of the OHPB LP for each media file m. These LP's are coupled by the constraint in (16), $\sum_m K_m = K$, which reflects the constrained total server bandwidth.

4.2 Solving OHPB-C

General linear programs with integer variables are NP-hard to solve. If every integer variable has a bounded finite domain, one may exhaustively enumerate all the possible value combinations of the integer variables. Exhaustively enumerating all value combinations for integer variables in OHPB-C is infeasible especially when we have a large number of media objects and a large number of server channels. Nonetheless, the particular problem structure of OHPB-C still permits a tailored solution algorithm design that runs more efficiently than general MILP approximation algorithms. An important observation is that integer variables in K_m are coupled in constraint (16) only, and with a feasible allocation of K, OHPB-C can be decomposed into a number of independent OHPB LPs, one for each broadcast session m.

A classic approximation technique for solving integer programs, and NP-hard problems in general, is *local search* [Bertsimas and Tsitsiklis 1997]. It searches the local "neighborhood" of the current solution, walks towards a better solution if existent, and terminates otherwise. In the context of OHPB-C, a neighborhood of a given channel allocation scheme includes all feasible allocations satisfying (16) that can be obtained from the current allocation by shifting a small number (e.g., 1) of channels between different sessions. We improve upon the general local search algorithm by incorporating a *guidance mechanism*, by selectively searching neighbors that are "promising". More specifically, we only consider neighbor allocations that assign more channels to the session that constitutes a bottleneck in optimizing the objective function. Our guided local search algorithm (for function M3) is presented in Figure 8.

We evaluated how our proposed approximation algorithm fared in terms of finding the optimal server bandwidth allocation. In test cases where we could obtain the optimal allocation by solving all possible allocations of K channels among the M media objects (e.g., $K \leq 10$, $M \leq 3$), we found that our approximation algorithm always yielded the optimal solution. Furthermore, we evaluated the sensitivity of the proposed algorithm to the initial allocation of channels (step 1). Specifically, for the experiments in Figure 9 we found that random initial allocation of channels among the media objects did not result in differing final channel allocation in any of the cases we considered.

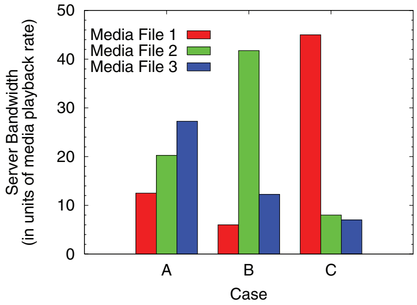


Fig. 9. Optimal channel allocation among three media files (model 3, $B = 60$, $r = 0.25$, and $M = 3$).

4.3 Numerical Results

The solution algorithm for OHPB-C presented in the preceding section was used to study how server bandwidth should be allocated among concurrent broadcasts of popular media files. We now present our empirical results and point out the key insights observed. The results presented here are for objective function $M3$.

We consider allocation of server channels among three concurrent broadcasts. The total server bandwidth $B = 60$ is partitioned into $K = 240$ channels, with channel transmission rate $r = B/K = 0.25$. Unless stated otherwise, it is assumed that each media file m has playback duration $L_m = 1$, there are 7 client types with bandwidths b_{ij}'s [1, 1.5, 2, 2.5, 3, 3.5, 4], and for each media file the 7 client types are assigned equal weights, $w_{ij} = 1/7$. For this configuration, referred to in the following as the baseline configuration, the optimal strategy is equal server bandwidth allocation to the three media files. Figure 9 explores three cases that examine the effect of heterogeneity in media playback durations, client weights, and client bandwidths respectively on server bandwidth allocation as discussed next.

In Case A, the effect of heterogeneous playback durations on server bandwidth allocation is examined by setting $L_1 = 1$, $L_2 = 2$, and $L_3 = 3$; all other conditions are identical to the baseline configuration. The number of channels allocated to each media file is roughly proportional to its playback duration. The underlying intuition is that a longer movie needs more server bandwidth support in an otherwise identical broadcast setting. Allocation of more channels to the longer media file enables this media file's clients to achieve start-up delays similar to that for the shorter media file. If the start-up delays in the objective function are normalized by media playback duration, giving an objective function of $\sum_j w_{mj}\tau_{mj}/L_m$ instead of $\sum_j w_{mj}\tau_{mj}$, smaller differences in the server bandwidth allocations are observed.

In Case B, only the receiver weights associated with the media file broadcasts are changed with respect to the baseline configuration. Specifically, the weight vectors associated with the first two media files are [0.1/6, 0.1/6, 0.1/6, 0.1/6, 0.1/6, 0.1/6, 0.9] and [0.9, 0.1/6, 0.1/6, 0.1/6, 0.1/6, 0.1/6, 0.1/6], respectively. In this scenario, the second media file receives the largest share of the server bandwidth, while the first media file receives the smallest share. This is due to the fact that for the second media file a high weight (0.9) is placed on the start-up delay of the lowest bandwidth clients, whereas for the first media file a high weight (0.9) is placed on the start-up delay of the highest bandwidth client. As objective function M3 aims to minimize the maximum over the three media files of the weighted start-up delay for a media file, substantial reductions in the start-up delay of low-bandwidth clients require relatively large increases in the server bandwidth allocation.

In Case C, each media file is accessed by a separate set of 7 client types. The bandwidths of the 7 client types accessing the first media file are as in the baseline configuration. Those for the client types

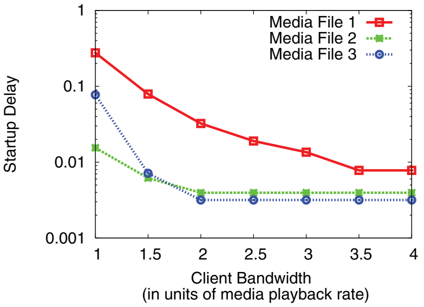


Fig. 10. Client start-up delays in case B (model 3, $B = 60$, $r = 0.25$, and $M = 3$).

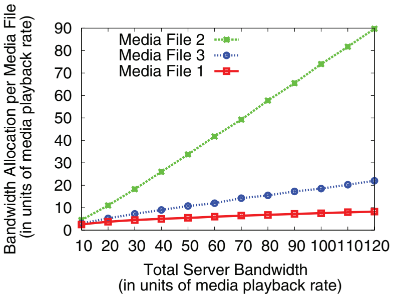


Fig. 11. Media file bandwidth allocations for varying total server bandwidth (model 3, $r = 0.25$, $M = 3$, Case B).

accessing the second media file are as in the baseline configuration, but each increased by 1, while those for the client types accessing the third media file are each increased by 2. In this case, the first media file receives the largest share of the server bandwidth, owing to the lower bandwidths of the client types that access it. This case together with the previous cases illustrates that with OHPB-C model 3, the server bandwidth allocation to each media file should be roughly proportional to its playback duration and to the weighted average bandwidth of its associated clients.

Figure 10 shows the start-up delay for each client type and media file, for Case B. Compared to the start-up delays for the third media file, the start-up delays for the first and second media files are optimized more towards high-bandwidth and low-bandwidth clients, respectively, because of the skewed weights of the client types for these files. Note that although the delay curve for the first media file lies entirely above that for the second media file, the *weighted* average client start-up delays are indeed equal for both files.

Figure 11 depicts server bandwidth allocation among the three media files (i.e., $K_1 \times r$, $K_2 \times r$, and $K_3 \times r$) as a function of the total server bandwidth, for Case B. It is interesting to observe that the increase of K_2 is superlinear, while the increases of K_1 and K_3 are sublinear. The explanation can be traced back to results in Figure 4; with a low weighted average client bandwidth, the weighted start-up delay decreases extremely slowly as server bandwidth is increased.

5. QUALITY ADAPTATION

This section describes techniques for quality adaptation using work-ahead when the client reception rate is time-varying and the media file has a cumulative layered encoding. Section 5.1 describes an approach for achieving work-ahead in OHPB systems. Client-side policies for performing work-ahead are considered in Section 5.2. Section 5.3 describes candidate rules for adding and dropping layers. A performance study of the resulting policy is presented in Section 5.4.

5.1 Efficient Work-Ahead

In a periodic broadcast system, the server transmits each segment at fixed rate to possibly multiple clients, and thus work-ahead cannot be achieved for a particular client by varying the segment transmission rate. Instead, a possible approach for achieving work-ahead is to download segments ahead of their scheduled download times.

Accomplishing work-ahead in this manner is complicated by the cyclic transmission of segments. As an example, consider a client that has partially received a segment, and then stops listening to the channel broadcasting this segment due to a drop in the reception bandwidth. If the client later resumes reception on the channel in time to receive data equivalent in amount to the data it is missing, it will be able to receive the remainder of the segment only if its reception period aligns with the broadcast of the missing portion.

A remedy to the given problem is to broadcast each segment as a digital fountain [Byers et al. 1998; Rizzo and Vicisano 1997] wherein erasure codes are applied to each segment so that a channel transmits a very long, potentially unbounded, sequence of encoded packets instead of transmitting the packets in a cyclic fashion. With erasure codes, all packets are essentially equivalent, and a segment can be reconstructed from any subset of packets of total size equal to (or possibly slightly greater than) the size of the segment. (We refer the reader to [Mitzenmacher 2004] for a survey of erasure codes that can facilitate digital fountains.) In previous work, the Reliable Periodic Broadcast protocols [Mahanti et al. 2003; Mahanti 2004] used erasure codes to enable efficient packet loss recovery. Here erasure codes are applied to achieve efficient work-ahead.

5.2 Work-Ahead Policy

The work-ahead policy determines how the reception bandwidth is allocated among the layers to provide maximal protection against short-term bandwidth fluctuations. We identify two considerations for work-ahead: the aggressiveness of the policy (i.e., with respect to attempting to maximize quality), and the allocation of work-ahead among the layers.

The aggressiveness of the policy is important because if a policy is too aggressive, it risks having many fluctuations in quality, which may be displeasing to the viewer [Zink et al. 2003]. If a policy is not aggressive enough, quality may increase very slowly, if at all, when additional bandwidth becomes available. Note that OHPB allows clients to trade off start-up delay for quality. The chosen start-up delay τ determines the bandwidth requirement for each received layer, and, therefore, the number of channels the client must concurrently listen to for each layer. By working ahead on a layer, it may be possible to sustain playback of the layer in the event that a temporary drop in reception bandwidth requires the client to listen to fewer of the layer's channels than required. In our proposed work-ahead policy, a client attempts to ensure that at least $T + \tau$ time units of media data are buffered for each layer, where T is a policy parameter. The work-ahead policy becomes more conservative as T is increased.

The allocation of work-ahead among layers is another important consideration and the following can be noted. First, work-ahead on lower layers is "safer" because it reduces the chances of work-ahead data being wasted in the event of a layer being dropped from playback [Rejaie et al. 1999a]. Note that when a layer is dropped, the data buffered for that layer becomes useless, and, with cumulative layering, all higher layers must be dropped from playback as well. Second, spreading the work-ahead among many layers allows the client to tolerate greater short-term bandwidth reduction. This is because, regardless of the amount of work-ahead that has been achieved on a layer, the bandwidth consumption of that layer can only be reduced to zero.

Suppose that a client is currently using n layers for playback and for d layers the buffering target is achieved (i.e., at least $T+\tau$ time units of media data are buffered). Taking the above into consideration, the following work-ahead rule is proposed: if there is reception bandwidth in excess of that needed to receive the currently subscribed layers, and $d < n$, the excess reception bandwidth is used to work-ahead on layers for which the buffering target is not achieved. The extra bandwidth is shared among these layers using a round robin scheme with a time quantum (as measured by the amount of achieved work-ahead) of T/n, to enable all layers to have a fair chance at getting T time units buffered. The order of round robin service begins with the lowest layer for which the buffering target is not achieved.

5.3 Policy for Adding/Dropping Layers

The decision to add a layer to the playback may depend on the current reception bandwidth, the currently achieved work-ahead, the bandwidth requirements of the currently subscribed layers and of the new layer, and the estimated future reception bandwidth. Obtaining a reasonable estimate of the future reception bandwidth may, however, be quite difficult. Thus, in previous work [Rejaie et al. 1999a] and in this work, the decision to add a new layer is taken when the instantaneous bandwidth exceeds the total bandwidth requirement of the subscribed layers in addition to the new layer, provided some work-ahead condition is satisfied.

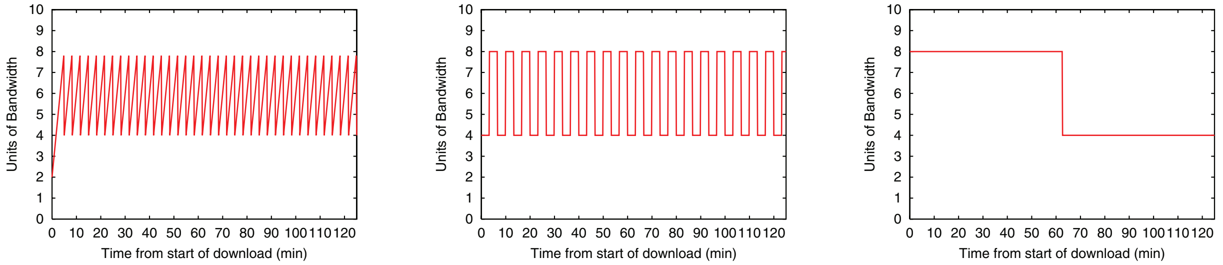
The required bandwidth for a layer can be determined as follows. First, note that unlike unicast streaming, in an OHPB system the bandwidth requirement typically exceeds the layer bit rate since clients must listen to multiple server channels and buffer data in advance of playout. Let b_l^τ denote the required client bandwidth for layer l for a start-up delay τ, given a particular setting of OHPB protocol parameters. Note that b_l^τ decreases once there are fewer than s_l^τ segments of the layer that are not yet completely received, and is given by $\min(K-k+1, s_l^\tau)\times r$, in units of the layer bit rate, where k denotes the index of the earliest such segment, and s_l^τ denotes the value of s that a client must use to achieve a start-up delay of τ.

Our approach is to add a layer when the available bandwidth exceeds that required to sustain the current subscription and the new layer, while receiving at twice the required rate b_l^τ on each of the $(n-d)$ layers for which the work-ahead buffering target is not achieved. Note that devoting additional bandwidth to a layer results in diminishing returns, since work-ahead bandwidth can only be used for retrieval of later segments not already being received, rather than for speeding up the retrieval of the segments whose reception is already in progress and that will be needed soon. The specific choice of twice the required rate was found to achieve an effective compromise between the goals of attaining the buffering targets and of making efficient use of reception bandwidth.

When bandwidth drops below the requirements of the work-ahead policy, the policy tries several strategies to reduce the amount of bandwidth being used. First, it will stop work-ahead on layers, starting with the highest quality layers. If this does not reduce bandwidth requirements enough, the policy will then reduce the reception rate for the highest quality layer (below the required rate b_l^τ), if necessary reducing this rate to zero, in which case the reception rate of the next highest quality layer may be reduced, and so on. This choice of reducing the reception rate on a layer to zero before reducing the rate on lower layers is motivated by the fact that data received from a layer may become useless if the layer is dropped from the playback, as may be required if work-ahead buffering is not sufficient to mask a long period of low client reception bandwidth. A layer is dropped if one of its segments has not been downloaded in time for playback.

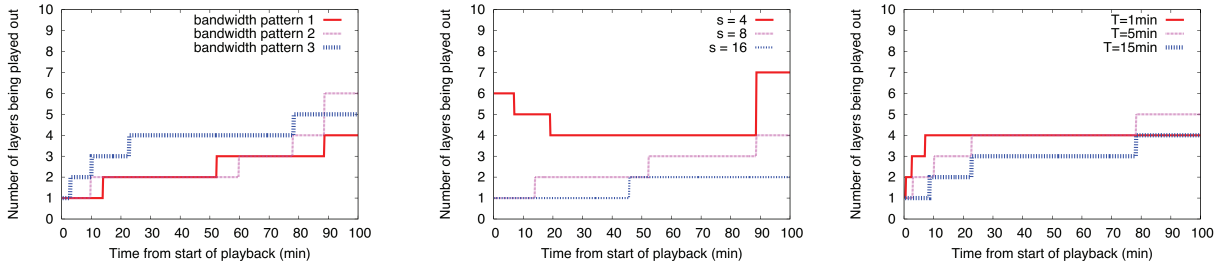
5.4 Performance Evaluation

This section presents sample evaluations of our quality adaptation policy assuming a 10 layer media file, where each layer is delivered using OHPB (model 2) with the same parameters, namely $B = 10$,



(a) Bandwidth Pattern 1 (b) Bandwidth Pattern 2 (c) Bandwidth Pattern 3

Fig. 12. Bandwidth patterns used for quality adaptation evaluation.



(a) Effect of Bandwidth Variation (s = 8, T = 5 min.) (b) Effect of Start-up Delay (T = 5 min., Bandwidth Pattern 1) (c) Effect of T (s = 8, Bandwidth Pattern 3)

Fig. 13. Example quality adaptation results.

$r = 0.25$, and client bandwidths [1, 1.5, 2, 2.5, 3, 3.5, 4], with equal weight assignment per client type. Simulations were run on various bandwidth variation patterns. The bandwidth patterns show available bandwidth, in units of media bit rate, as a function of time. Each bandwidth pattern is designed to test different aspects of the quality adaptation algorithm. A sawtooth bandwidth variation pattern (Figure 12(a)) is used to model bandwidth fluctuations that may be seen owing to application of TCP-like or TCP-friendly congestion control algorithms. The second bandwidth variation pattern (Figure 12(b)) incorporates many increases and decreases in bandwidth. However, in this pattern, the increases and decreases are sharper than in the first bandwidth variation pattern. Because layer addition/removal causes bandwidth usage to change in multiples of $s * r$, the work-ahead policy should show improved bandwidth utilization with the second bandwidth variation pattern. We also report simulation results that test the performance of the work-ahead policy under exceptional circumstances such as a major bandwidth decrease halfway through the client session (Figure 12(c)). The goal was to measure how much quality is preserved in such a case. We have experimented with other bandwidth patterns (e.g., inverse of bandwidth pattern 3, variations to bandwidth patterns 1 and 2), but for brevity, restrict our discussion here to the above-mentioned bandwidth patterns; experiments with these bandwidth patterns yielded qualitatively similar results.

The performance of the quality adaptation policy under various bandwidth conditions is examined in Figure 13(a), for clients that choose their start-up delay assuming that they will be able to achieve a reception rate of 2 (in units of the media playback rate) per received layer, and that begin cautiously with only the base layer. The policy is able to exploit bandwidth that is available during the sawtooth and square wave fluctuations. The work-ahead policy uses this extra bandwidth to buffer data from segments occurring later in the media file, and as a result is able to deliver higher quality for the latter portion of the playback. As predicted, higher levels of quality are achieved with the second, square wave bandwidth variation pattern, than with the first pattern. With the third bandwidth variation pattern, a large amount of work-ahead is achieved over the first 60 minutes of the client session, and therefore the work-ahead policy is able to sustain a high level of quality following the subsequent abrupt drop in bandwidth.

Figure 13(b) examines the impact of the per-layer client reception bandwidth assumed when clients choose their start-up delay, for the first bandwidth variation pattern. Since $r = 0.25$ in these experiments, $s = 4$, $s = 8$, and $s = 16$ correspond to per-layer reception bandwidths of 1, 2, and 4, respectively. Note that the case of $s = 16$ corresponds to selection of a very short start-up delay, at the cost of a high required reception bandwidth per layer, implying a relatively low quality playback. With $s = 4$, in contrast, the corresponding start-up delay is quite long, and by the time playback begins the work-ahead policy has already added 5 additional layers for playback beyond the base layer (although such a high level of quality cannot be sustained). In all cases, the work-ahead policy is able to achieve sufficient work-ahead to allow quality to be increased for a latter portion of the playback.

As stated previously, the value of T determines how aggressively the protocol will behave when adding layers. The effects of T are examined with bandwidth pattern 3 in Figure 13(c), for $T = 1$ minute, $T =$ 5 minutes, and $T = 15$ minutes. The third bandwidth pattern is chosen because an objective of the work-ahead buffering target as parameterized by T is to provide high likelihood that playback quality can be sustained when bandwidth drops for extended periods of time. In each case, the percent of received data that is not used owing to the respective layer being dropped from playback, is also measured. It is found that this occurs only for $T = 1$ minute; in this case, 0.69% of the received data is wasted. This suggests that when T is low, the client attempts to add many layers but when bandwidth drops, it does not have enough buffered data on these layers to download partially completed segments in time for playback. Intuitively, when $T = 15$ minutes the client slowly adds layers and does not achieve higher levels of quality. In these evaluations, $T = 5$ minutes yields the best performance, providing an effective balance between the desire for increased quality, and protection against layer drops as provided by more substantial work-ahead.

6. CONCLUSIONS

This article has addressed the challenge of streaming popular media files, on-demand, to heterogeneous clients. We considered both heterogeneity in the sense of different clients having different reception bandwidths, and heterogeneity due to time-varying client reception bandwidth. The new Optimized Heterogeneous Periodic Broadcast (OHPB) protocol that is developed supports heterogeneous client bandwidths better than previous periodic broadcast protocols, and provides a tunable degree of differentiation between clients types with differing achievable reception rates, with respect to their associated start-up delays. A novel methodology, based on linear programming models, is used to develop the OHPB segment size progression. We also developed a generalization of the OHPB linear optimization model, OHPB-C, that allows optimal server bandwidth allocation among multiple concurrent OHPB broadcasts, wherein each media file and its clients may have different characteristics. For delivery of layered media files using OHPB, efficient quality adaptation policies are developed that allow each client to independently determine how to best allocate its time-varying reception bandwidth at each instant of time, so as to achieve more uniform and higher quality playback.

APPENDIX

A. Subgradient Algorithm for the OHPB LP

To construct the subgradient algorithm, we first reformulate the OHPB LP as follows. Observe that constraint groups (6)–(8) essentially impose inequality relations between τ_j and linear combinations of l_i. By successive substitutions of (7)–(8) into (6), we obtain the revised OHPB LP, which is equivalent to the original:
Minimize

$$\sum_j w_j \tau_j$$

subject to:

$$\begin{aligned}\sum_i l_i &= L\\ \tau_1 &\geq \alpha_{11}^{(1)} l_1 + \alpha_{11}^{(2)} l_2 + \cdots + \alpha_{11}^{(K)} l_K\\ &\cdots\\ \tau_1 &\geq \alpha_{1K}^{(1)} l_1 + \alpha_{1K}^{(2)} l_2 + \cdots + \alpha_{1K}^{(K)} l_K\\ &\cdots\\ \tau_m &\geq \alpha_{m1}^{(1)} l_1 + \alpha_{m1}^{(2)} l_2 + \cdots + \alpha_{m1}^{(K)} l_K\\ &\cdots\\ \tau_m &\geq \alpha_{mK}^{(1)} l_1 + \alpha_{mK}^{(2)} l_2 + \cdots + \alpha_{mK}^{(K)} l_K\\ l_i, \tau_j &\geq 0, \quad \forall i, j.\end{aligned}$$

We next derive an equivalent Lagrange dual problem of the above LP, by relaxing all the lower-bound constraints on τ. We introduce corresponding dual prices λ_{ij}, and modify the objective function as follows:
Minimize

$$\sum_j w_j \tau_j + \sum_i \sum_j \left[\sum_k \alpha_{ij}^{(k)} l_i - \tau_i \right]$$

subject to:

$$\mathcal{P}_1 : \begin{cases} \sum_i l_i = L \\ l_i, \tau_j \geq 0 \quad \forall i, j. \end{cases}$$

By Lagrange duality, an optimal solution to the relaxed LP is always an upper bound for the optimal solution of the original OHPB LP; furthermore, by varying dual prices, the maximum optimal solution to the relaxed LP exactly equals the optimal solution to the OHPB LP. Therefore, we can focus on the following Lagrange dual problem instead:

$$\max_{\lambda_{ij} \geq 0} \min_{l \in \mathcal{P}_1} \left[\sum_j w_j \tau_j + \sum_i \sum_j \left(\lambda_{ij} \left(\sum_k \alpha_{ij}^{(k)} l_i - \tau_i \right) \right) \right] \tag{19}$$

$$= \max_{\lambda_{ij} \geq 0} \min_{l \in \mathcal{P}_1} \left[\sum_i \left(w_i - \sum_j \lambda_{ij} \right) \tau_i + \sum_i \sum_j \left(\lambda_{ij} \sum_k \left(\alpha_{ij}^{(k)} l_i \right) \right) \right]. \tag{20}$$

Note that dual feasibility requires $\sum_j \lambda_{ij} \leq w_i$, because otherwise the inner minimization is unbounded. Therefore, (20) is equivalent to:

$$\max_{\lambda \in \mathcal{P}_2} \min_{l \in \mathcal{P}_1} \sum_i \sum_j \left(\lambda_{ij} \sum_k \left(\alpha_{ij}^{(k)} l_i \right) \right), \tag{21}$$

where $\mathcal{P}_2$ is the following feasibility polytope of vector λ:

$$\mathcal{P}_2 : \begin{cases} \sum_j \lambda_{ij} \leq w_i & \forall i \\ \lambda_{ij} \geq 0 & \forall i, j. \end{cases}$$

The subgradient algorithm for (21) starts with an initial vector λ. It iteratively updates the primal vector l and the dual vector λ until convergence. Any feasible vector in $\mathcal{P}_2$ can be used to initialize λ, e.g., $\lambda_{ij} = w_i/K, \forall i, j$. Then in each iteration of the subgradient algorithm, we first update l, by assuming λ as a constant vector, and solve the inner minimization problem $\min_{l \in \mathcal{P}_1} \sum_i \sum_j (\lambda_{ij} \sum_k (\alpha_{ij}^{(k)} l_i))$. This sub-problem has a nice combinatorial structure, and can be efficiently solved. Let β be the following constant vector:

$$\beta_i = \sum_j \left(\lambda_{ij} \sum_k \alpha_{ij}^{(k)} \right), \qquad \forall i.$$

The minimization problem above is reformulated into:

$$\min_{l \in \mathcal{P}_1} \sum_i \beta_i l_i.$$

Let $i^* = \text{argmin}_i \beta_i$, then the optimal vector l can be computed as follows:

$$l_i = \begin{cases} 0 & \forall i \neq i^* \\ L & i = i^*. \end{cases}$$

We next update the dual price vector λ based on values in l and a prescribed sequence of step sizes in vector θ:

$$\lambda'_{ij} = \lambda_{ij}[k] + \theta[k] \sum_t \left(\alpha_{ij}^{(t)} l_t \right).$$

λ' is in general infeasible; it is projected into the feasible polytope $\mathcal{P}_2$ and then becomes the new value for λ in the next iteration:

$$\lambda_{ij}[k+1] = \frac{\lambda'_{ij}}{\sum_j \lambda'_{ij}} w_i.$$

After both primal and dual variables are updated, the next iteration of the subgradient algorithm starts. As long as the step size sequence in θ satisfies $\theta[k] \geq 0, \lim_{k \to \infty} \theta[k] = 0$ and $\sum_k \theta[k] = \infty$, the algorithm converges at optimal values in λ. Then primal recovery techniques can be applied to obtain the optimal vector l^*, through a convex combination of intermediate values of l computed along the way of convergence [Sherali and Choi 1996].

REFERENCES

ALMEIDA, J., EAGER, D., FERRIS, M., AND VERNON, M. 2002. Provisioning content distribution networks for streaming media. In *Proceedings of the IEEE INFOCOM*.

BAR-NOY, A., GOSHI, G., LADNER, R., AND TAM, K. 2002. Comparison of stream merging algorithms for media-on-demand. In *Proceedings of the International Conference on Multimedia Computing and Networking*.

Bertsimas, D. and Tsitsiklis, J. N. 1997. *Introduction to Linear Optimization*. Athena Scientific, Nashua NH.

Byers, J. and Kwon, G. 2001. STAIR: A practical AIMD multirate multicast congestion control. In *Proceedings of the International Workshop on Networked Group Communication*.

Byers, J., Luby, M., Mitzenmacher, M., and Rege, A. 1998. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the ACM SIGCOMM*.

Cheung, S., Ammar, M., and Li, X. 1996. On the use of destination set grouping to improve fairness in multicast video distribution. In *Proceedings of the IEEE INFOCOM*.

Eager, D., Ferris, M., and Vernon, M. 1999. Optimized regional caching for on-demand data delivery. In *Proceedings of the International Conference on Multimedia Computing and Networking*.

Eager, D., Ferris, M., and Vernon, M. 2000. Optimized caching in systems with heterogeneous client populations. *Performance Evaluation (Special Issue on Internet Performance Modeling) 42*, 2/3, 163–185.

Eager, D. and Vernon, M. 1998. Dynamic skyscraper broadcasts for video-on-demand. In *Proceedings of the International Workshop on Advances in Multimedia Information Systems*.

Eager, D., Vernon, M., and Zahorjan, J. 1999. Optimal and efficient merging schedules for video-on-demand servers. In *Proceedings of the ACM Multimedia Conference*.

Eager, D., Vernon, M., and Zahorjan, J. 2000. Bandwidth skimming: A technique for cost-effective video-on-demand. In *Proceedings of the International Conference on Multimedia Computing and Networking*.

Eager, D., Vernon, M., and Zahorjan, J. 2001. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. Knowl. Data Engin. 13*, 5, 742–757.

Floyd, S. and Fall, K. 1999. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. Netw. 7*, 4, 458–472.

Gao, L., Kurose, J., and Towsley, D. 1998. Efficient schemes for broadcasting popular videos. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

Hu, A. 2001. Video-on-demand broadcasting protocols: A comprehensive study. In *Proceedings of the IEEE INFOCOM*.

Hu, A., Nikolaidis, I., and Beek, P. 1999. On the design of efficient video-on-demand broadcast schemes. In *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simalation of Computer and Telecommunication Systems*.

Hua, K. and Sheu, S. 1997. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of the ACM SIGCOMM*.

Hua, K. A., Bagouet, O., and Oger, D. 2003. Periodic broadcast protocol for heterogeneous receivers. In *Proceedings of the International Conference on Multimedia Computing and Networking*.

Juhn, L. and Tseng, L. 1997. Harmonic broadcasting for video-on-demand service. *IEEE Trans. Broadcast. 43*, 3, 268–271.

Kim, T. and Ammar, M. 2001. A comparison of layering and stream replication video multicast schemes. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

Legout, A. and Biersack, E. 2000. PLM: Fast convergence for cumulative layered multicast transmission schemes. In *Proceedings of the ACM SIGMETRICS*.

Li, X. and Ammar, M. H. 1996. Bandwidth control for replicated-stream multicast video distribution. In *Proceedings of the IEEE International Symposium on High-Performance Distributed Computing*.

Li, X., Ammar, M. H., and Paul, S. 1999. Video multicast over the Internet. *IEEE Netw. 13*, 2, 46–60.

Luby, M., Goyal, V., Skaria, S., and Horn, G. 2002. Wave and equation based rate control using multicast round trip time. In *Proceedings of the ACM SIGCOMM*.

Mahanti, A. 2004. Scalable Reliable On-demand Media Streaming Protocols. Ph.D. thesis, Department of Computer Science, University of Saskatchewan.

Mahanti, A., Eager, D., and Vernon, M. 2005. Improving multirate congestion control using a TCP Vegas throughput model. *Comput. Netw. 48*, 2, 113–136.

Mahanti, A., Eager, D., Vernon, M., and Sundaram-Stukel, D. 2003. Scalable on-demand media streaming with packet loss recovery. *IEEE/ACM Trans. Netw. 11*, 2 (April), 195–209.

McCanne, S., Jacobson, V., and Vetterli, M. 1996. Receiver-driven layered multicast. In *Proceedings of the ACM SIGCOMM*.

Mitzenmacher, M. 2004. Digital fountains: A survey and look forward. In *Proceedings of the IEEE Information Theory Workshop*.

Paris, J., Carter, S., and Long, D. 1998. Efficient broadcasting protocols for video-on-demand. In *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simalation of Computer and Telecommunication System*.

Qudah, B. and Sarhan, N. J. 2006. Towards scalable delivery of video streams to heterogeneous receivers. In *Proceedings of the ACM Multimedia Conference*.

Rejaie, R., Handley, M., and Estrin, D. 1999a. Quality adaptation for congestion controlled video playback over the Internet. In *Proceedings of the ACM SIGCOMM*.

Rejaie, R., Handley, M., and Estrin, D. 1999b. RAP: An end-to-end congestion control mechanism for realtime streams in the Internet. In *Proceedings of the IEEE INFOCOM*.

Rizzo, L. and Vicisano, L. 1997. A reliable multicast data distribution protocol based on software FEC techniques. In *Proceedings of the IEEE Workshop on High Performance Communication Systems*.

Sen, S., Rexford, J., and Towsley, D. 1999. Proxy prefix caching for multimedia streams. In *Proceedings of the IEEE INFOCOM*.

Sherali, H. and Choi, G. 1996. Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Oper. Resea. Lett. 19*, 5, 105–113.

Tantaoui, M., Hua, K., and Do, T. 2004. BroadCatch: A periodic broadcast technique for heterogeneous video-on-demand. *IEEE Trans. Broadcast 50*, 3, 289–301.

Turletti, T., Parisis, S., and Bolot, J. 1997. Experiments with a layered transmission scheme over the Internet. INRIA Tech. rep. 3296.

Vicisano, L., Rizzo, L., and Crowcroft, J. 1998. TCP-like congestion control for layered video multicast data transfer. In *Proceedings of the IEEE INFOCOM*.

Viswanathan, S. and Imielinski, T. 1996. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimed. Syst. 4*, 4, 197–208.

Wang, B., Sen, S., Adler, M., and Towsley, D. 2002. Optimal proxy cache allocation for efficient streaming media distribution. In *Proceedings of the IEEE INFOCOM*.

Widmer, J., Dende, R., and Mauve, M. 2001. A survey of TCP-friendly congestion control. *IEEE Netw. 15*, 3, 28–37.

Zhao, Y., Eager, D., and Vernon, M. 2002. Network bandwidth requirements for scalable on-demand streaming. In *Proceedings of the IEEE INFOCOM*.

Zink, M., Künzel, O., Schmitt, J., and Steinmetz, R. 2003. Subjective impression of variations in layer encoded videos. In *Proceedings of the IEEE International Workshop on Quality of Service*.

Received April 2007; revised August 2007; accepted November 2007.