Scalable Streaming for Heterogeneous Clients

Liqi Shi
Dept. Elec. & Comp. Eng.
University of Calgary
Calgary, Canada
lishi@ucalgary.ca

Phillipa Sessini
Dept. Comp. Sci.
University of Calgary
Calgary, Canada
psessini@ucalgary.ca

Anirban Mahanti
Dept. Comp. Sci.
University of Calgary
Calgary, Canada
mahanti@cpsc.ucalgary.ca

Zongpeng Li
Dept. Comp. Sci.
University of Calgary
Calgary, Canada
zongpeng@cpsc.ucalgary.ca

Derek L. Eager
Dept. Comp. Sci.
University of Saskatchewan
Saskatoon, Canada
eager@cs.usask.ca

ABSTRACT

Periodic broadcast protocols enable the 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 available bandwidth is not time-varying. In this paper, 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 propose complementary client protocols employing work-ahead buffering of data during playback, so as to enable more uniform playback quality when the available 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

Keywords: Scalable Streaming, Periodic Broadcasts, Linear Programming, Quality-of-Service

1. INTRODUCTION

Media-on-Demand systems that utilize *periodic broadcast* protocols can concurrently serve large numbers of clients with low startup delay using fixed server resources. The past decade saw the development of a number of periodic broadcast protocols such as Pyramid Broadcasting [28], Harmonic Broadcasting [13], Skyscraper Broadcasting [11], Quasi Harmonic Broadcasting [22], Dynamic Skyscraper [4], and more recently, the Optimized Periodic Broadcast protocol family [20]. While the aforementioned protocols achieve differing tradeoffs, for example between startup delay and server bandwidth, they all operate within a common framework. 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 pre-determined schedule. Clients requesting a media file are given a startup delay to begin playback and a schedule for receiving segments from the server channels that guarantees 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) [12] and BroadCatch [26] protocols which we discuss later in the paper, 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. 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 [7]. 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.

One strategy for accommodating heterogeneous client reception rates is to use either a *simulcast* [3, 14] approach or a *layered* media encoding [17, 21] approach. With the simulcast approach different versions of the media file are encoded at different bit rates, and a selected periodic broadcast protocol may be used to deliver each version. 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 available bandwidth can help clients determine which layers/version to receive. Using layering or simulcasting with a periodic broadcast scheme tailored for a single bandwidth only partially addresses the problem. Specifically, such solutions do *not* allow efficient tradeoffs between startup delay and media quality. For example, a client with slightly lower (average) bandwidth than that required to receive a media of desired quality (i.e., layers or version) requires a relatively large increase in the startup delay to guarantee continuous playback of the higher quality content. In addition, time-varying available bandwidth due to use of a multirate congestion control algorithm [15, 16, 17, 18, 19, 27], for example, further compounds the problem. For the case of layered media files (or replicated streams), 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 available client reception bandwidth, and heterogeneity due to time-varying available bandwidth. 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, for delivery of layered media files, we develop complementary *quality adaptation* [23] mechanisms and policies that allow each client to independently determine how to allocate time-varying available 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 result in different startup delays at a client with a certain reception bandwidth. For systems with homogeneous clients, this delay optimization problem has been effectively addressed [20]. However, designing an optimal progression for a population of clients with heterogeneous reception bandwidth is a new problem yet to be investigated. The challenge lies essentially in making good tradeoffs between startup delays experienced by the different clients. In this paper, we model such an optimization problem as a linear program, and discuss an efficient subgradient algorithm for solving this linear program.

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 owing 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 that 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 and evaluation 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* [2] to avoid the aforementioned problems with cyclic segment transmissions.

The remainder of this paper 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. Our quality adaptation proposal is presented in Section 4. Conclusions are presented in Section 5.

2. BACKGROUND

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. 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 [6, 9, 29].

Among the scalable multicast streaming protocols, the *periodic broadcast* [1, 8, 11, 13, 20, 28] protocols and the *stream merging* [5, 6] protocols are the most effective. The 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 [20]. Our work uses this protocol as a building block because: 1) it can support any client reception rate, even reception rates 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, the Broadcatch [26] and the Heterogeneous Receiver-Oriented Broadcasting (HeRO) [12] protocols, respectively. Section 2.4 reviews prior work on quality adaptation for unicast streaming and discusses the difficulties that arise when considering 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; pe-

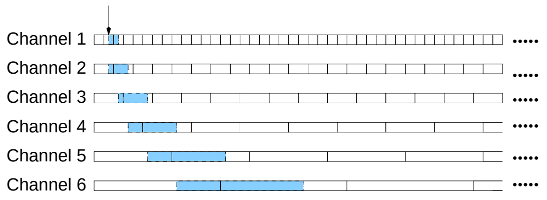


Figure 1: Optimized PB ($K = 6$, $r = 1$, $s = 2$)

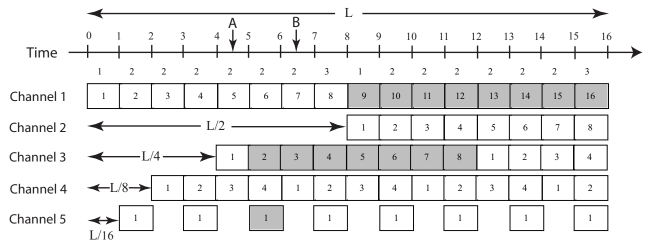


Figure 2: BroadCatch ($K = 5$, $r = 1$)

riods 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.

The Optimized PB protocol uses an optimal (in the sense of minimizing client startup 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 [26]. 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 length) 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 broadcast on channel 1.

Figure 2 shows 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, and 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 startup delay of clients. However, in most cases, clients can achieve reduced startup delay when increased server bandwidth is used for delivery of the file (i.e., K increases), only when the client reception rate also increases.

2.3 HeRO

The HeRO protocol is another technique for clients with heterogeneous bandwidths [12]. 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 may be 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. This increased frequency of broadcast of the large segments aids in reducing the startup latency of low bandwidth clients. As illustrated by the numerical results presented in Section 3.2, HeRO and BroadCatch have similar performance.

2.4 Quality Adaptation

The most closely related previous work on quality adaptation has considered the context of unicast streaming of layered media files [23]. In this work, a TCP-friendly rate adaptation policy [24] is utilized to determine the transmission rate of the server. During periods of high bandwidth, 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 periods of low bandwidth. Thus, the server adds or drops media layers in response to long-term changes in the available bandwidth, while work-ahead is used to smooth short-term variations in the available bandwidth.

With streaming applications that employ multirate congestion control [15, 16, 17, 18, 19, 27], it is not feasible to change the transmission rate on a channel to accommodate the needs of an individual client since 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 of 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 available bandwidth.

For periodic broadcast protocols such as Harmonic Broadcasting [13] and its variants [10, 22] in which clients listen concurrently to all of the server channels that are transmitting segments of the requested file, it is impossible to work-ahead unless the server somehow transmits additional streams for this purpose. With Optimized PB, BroadCatch, HeRO, or any other protocols in which clients listen to a subset of channels, a client that temporarily has extra band-

Table 1: Notation for the OHPB Protocol

Symbol	Definition
K	Total number of segments (server channels)
l_i	Playback duration of the i^{th} segment
L	Total media playback length
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 startup delay of type j clients
w_j	Weight used for clients of type j

width 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 available 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. Initially, it is assumed that the rate at which each client can receive media data does not change substantially during its session; this assumption is relaxed in Section 4. Section 3.1 develops the new OHPB protocol and outlines the OHPB linear program. This discussion is applicable for the delivery of a single monolithic media file or a single layer of a layer-encoded media file. Numerical results illustrating OHPB performance, and comparisons with HeRO and BroadCatch, are presented in Section 3.2. Possible tradeoffs between media quality and startup delay for layered media files are discussed in Section 3.3. We conclude this section with a qualitative discussion of OHPB's salient features. For ease of reference, Table 1 summarizes the notation used in the OHPB linear program.

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 length L, is partitioned into K segments; each segment i, $1 \leq i \leq K$, is of length 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 startup delay and a schedule for tuning into the channels. The startup 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 [20], and furthermore, allows design of flexible quality adaptation techniques as described in Section 4.

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 startup 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 under some 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 startup delays for the N types of client, yielding the following OHPB linear program (LP):

$$\text{Minimize} \qquad M1 = \textstyle\sum_j w_j \tau_j \tag{4}$$

Subject to:

$$\begin{cases} \sum_i l_i = L & (5) \\ t_j(k) \leq \tau_j + \sum_{i=1}^{k-1} l_i \quad \forall j, k & (6) \\ t_j(k) = \frac{l_k}{r} \quad \forall j, 1 \leq k \leq s_j, s_j \geq 1 & (7) \\ t_j(k) = t_j(k - s_j) + \frac{l_k}{r}, \quad \forall j, s_j < k \leq K, s_j \geq 1 & (8) \end{cases}$$

$$l_i, \tau_j, t_j(k) \geq 0, \qquad \forall i, j, k$$

The inputs to the OHPB LP are K, r, N, s_j's, w_j's, and L. The LP outputs the sequence of segment sizes (l_i's), and the startup 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 sub-gradient 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 paper, exploits the specific underlying structure of the OHPB LP and achieves much higher scalability than general simplex or interior-point algorithms.

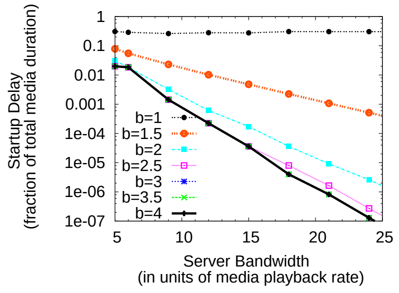


Figure 3: Scalability of OHPB (model 2, $r = 0.25$)

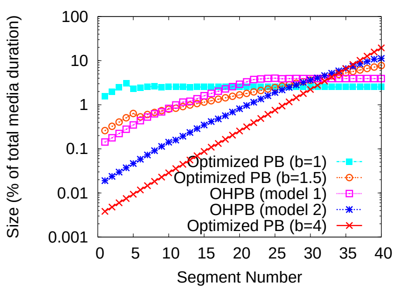


Figure 4: Segment Size Progressions ($B = 10$, $r = 0.25$)

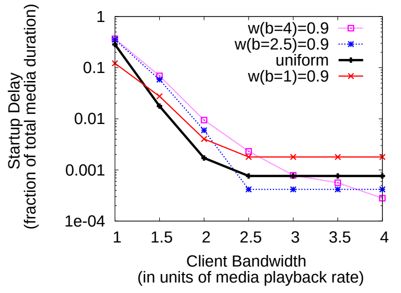


Figure 5: Adaptivity of OHPB (model 2, $B = 10$, $r = 0.25$)

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 startup delays for low bandwidth clients at the cost of increasing startup delays for high bandwidth clients. Alternatively, if clients with higher bandwidth make requests more often, a broadcast scheme to favour 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 startup 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 startup delay using an Optimized PB protocol tailored to achieve the minimum startup delay for client type j, given server bandwidth $B = K \times r$. The second objective function is then given as follows:

$$\text{Minimize} \qquad M2 = \sum_j w_j \frac{\tau_j}{\tau_j^{opb}}. \qquad (9)$$

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.

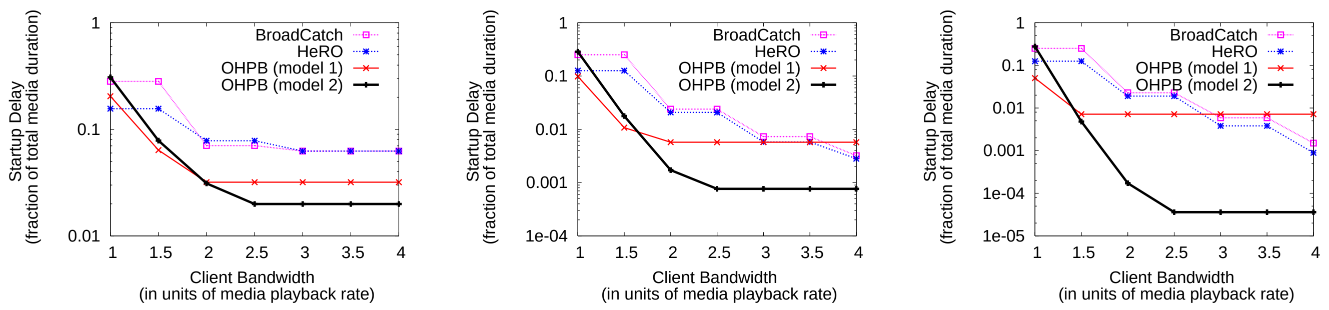
3.2 Numerical Results

Previous periodic broadcast protocols have the desirable property that *linear* increases in server bandwidth yield *exponential* decreases in startup delay [9]. A key question is whether or not this attractive property holds for OHPB systems designed to support clients with heterogeneous bandwidths.

Figure 3 shows the required startup delay for specific client bandwidths as a function of server bandwidth. 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 the aforementioned property largely holds for clients with data rates greater than the media playback rate (i.e., $b > 1$). For clients with bandwidth equal to the media playback rate, increasing server bandwidth has negligible impact on startup delay since 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. This is because the higher startup 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 4 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. For reference, the figure also shows the Optimized PB segment size progressions for $b = 1$, $b = 1.5$, and $b = 4$. Note that OHPB model 2 exhibits exponential increase in segment sizes and yields segment sizes most similar to the exponentially increasing 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 bandwidths also requires larger initial segments. From the figure, we observe that OHPB model 1 has significantly slower growth in segment sizes compared to OHPB model 2 because the former tries to lower the startup delays for low bandwidth clients at the cost of increased delays for the high bandwidth clients. In fact, the initial segments of OHPB model 1 are similar to Optimized PB ($b = 1.5$), while the later segments are similar to Optimized PB ($b = 1$).

Figure 5 explores the startup delay experienced by clients for different weight assignments. We report results obtained by solving the OHPB LP (model 2) for $B = 10$, $r = 0.25$, and client bandwidths $[1, 1.5, 2, 2.5, 3, 3.5, 4]$ with different weight assignments. In the figure, four example weight assignments for OHPB model 2 are considered, namely: 1) the client type with bandwidth $b = 1$ is assigned weight $9/10$, with the remaining types assigned equal weight of $1/60$; 2) same as (1) but with the client type with bandwidth $b = 2.5$ the one that is assigned weight $9/10$; 3) same as (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 startup 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 startup delay for these clients compared to when all client types have equal weight assignments. This decrease in startup delay, however, comes



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

Figure 6: Comparing OHPB, BroadCatch, and HeRO

at the cost of increased startup 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 3-5. Similar comparative results are obtained with other sets of bandwidth values. Figure 6 shows the startup 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 startup delay results for BroadCatch and HeRO are averages over all distinct time slots in the respective transmission schedule, of the startup delay for a client arriving in that time slot. Note that OHPB provides lower startup delay than both HeRO and BroadCatch for most client bandwidths. OHPB model 2, in particular, provides substantially lower startup 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 startup delay.

3.3 Startup 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 OHPB (model 2) with client bandwidths and other parameters identical to those used for Figure 6(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(a), it can be seen that the client has a choice of receiving all 10 layers with a startup delay of 9 minutes, 6 layers with a startup delay of 2 minutes, 5 layers with a startup delay of 56 seconds, or 4 layers with a startup delay of 36 seconds. There is no advantage to receiving less than 4 layers as it does not reduce the startup 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 startup delay of 18 minutes, 6 layers with a startup delay of 8 minutes, 5 layers with a startup delay of 3 minutes, or 4 layers with a startup delay of 27 seconds. Had the file been broadcast using the BroadCatch protocol (using $B = 5$ per layer as in Figure 6(a)), the client could receive 10 layers with a startup delay of 8 minutes, 5 layers with a startup delay of 2 minutes, or 3 layers with a startup delay of 1.9 minutes. Since BroadCatch cannot take advantage of fractional bandwidth, the layer/delay tradeoffs available using BroadCatch are somewhat limited. Clearly, OHPB offers a much better range of tradeoffs between media quality and startup delays.

3.4 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 startup delay of clients). Both BroadCatch and HeRO include no 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 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 approach [20] of delivering each segment as a *digital fountain* [2] 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, the aforementioned packet loss recovery mechanisms are not applicable. Finally, as discussed in the next section, OHPB can be extended to support quality adaptation when the achievable client reception bandwidth is time-varying.

4. QUALITY ADAPTATION

This section describes techniques for quality adaptation using work-ahead when the achievable client reception rate is time-varying, and the media file has a layered encoding. Section 4.1 describes an approach for achieving work-ahead in OHPB systems. Client-side policies for performing work-ahead are considered in Section 4.2. Section 4.3 describes candidate rules for adding and dropping layers. A performance study of the resulting policy is presented in Section 4.4.

4.1 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 time.

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 available 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 above problem is to apply erasure codes to each segment so that a channel transmits a very long 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 equal (or possibly slightly longer) in total size to the size of the segment. In previous work, the Reliable Periodic Broadcast protocols [20] used erasure codes to enable efficient packet loss recovery. Here we apply erasure codes to achieve efficient work-ahead.

4.2 Work-ahead Policy

The work-ahead policy determines how the available 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, 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 [30]. If a policy is not aggressive enough quality may increase very slowly, if at all. Note that OHPB allows clients to trade off startup delay for quality. The number of layers selected by a client determines the per-layer bandwidth requirement (and therefore, the number of channels the client must listen to concurrently on each layer), and the startup delay τ. We add a buffering constant, T, to the buffering requirements of OHPB, that is at any given point in the download, a layer attempts to have $T + \tau$ time units buffered. The buffering constant aims to provide a "soft" guarantee on the amount of time playback of a layer can be sustained in the event of a bandwidth drop. When T is small, this work-ahead policy behaves more aggressively, whereas when T is larger the policy is more conservative.

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" since it reduces the chances of work-ahead data being wasted in the event of a layer drop [23]; it is "safer" to work-ahead on lower layers because in the event of a layer drop, any work-ahead on that layer is lost, and consequently (in case of cumulative layering), the buffering on all layers above may become useless. 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 available on a layer, the bandwidth consumption of that layer can only be reduced to zero.

Suppose that a client is receiving n layers and on d layers the buffering constraint is satisfied (i.e., $T+\tau$ time units are buffered). Taking the above into consideration, the following work-ahead rule is proposed: if there is bandwidth in excess of that needed to receive the currently subscribed layers, and $d < n$, work-ahead on layers that do not satisfy the buffering requirement. The extra bandwidth is shared using a round robin scheme with a time quantum 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 that does not meet the buffering requirement.

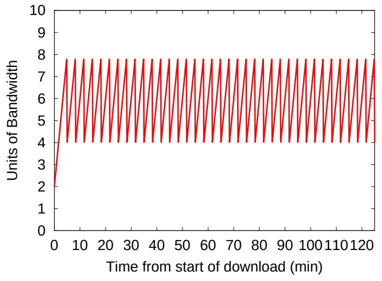
4.3 Policy for Adding/Dropping Layers

The decision to add a layer may depend on currently available bandwidth, currently achieved work-ahead, the bandwidth requirements of the currently subscribed layers and the new layer, and an estimate of the bandwidth available in the future. Obtaining a reasonable estimate of the future available bandwidth may, however, be quite difficult. Thus, in previous work [23] and in this work, the decision to add a new layer is taken when the instantaneous bandwidth exceeds the bandwidth requirement of the subscribed layers in addition to the new layer, provided some work-ahead condition is satisfied.

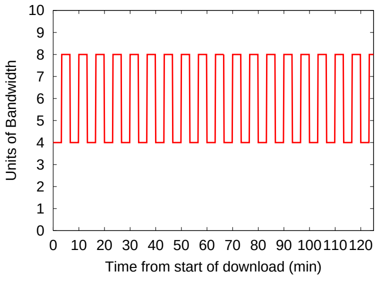
Determining the bandwidth required for the current subscription level needs some special consideration. Note that unlike unicast streaming, the bandwidth requirement in an OHPB system does not equal the layer bit rate as the clients are required to listen on multiple server channels at an aggregate rate greater than the layer bit rate. Furthermore, a client can choose a higher startup delay that enables it to subscribe to more layers, using a lower total reception rate on each layer. Let b_l^τ denote the required client bandwidth for layer l for a startup delay τ, as determined for a particular OHPB protocol setting (i.e., as described in Section 3.3). Note that b_l^τ decreases near the end of the download, once there are fewer than s_l^τ segments remaining to receive, and is given by $\min(K - k, s_l^\tau) \times r$, in units of the layer bit rate, where k is the earliest segment currently being downloaded on the layer, and s_l^τ is the value of s that a client must use to achieve a startup 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 continuing to buffer on the $(n-d)$ layers that do not meet the work-ahead requirements. The specific heuristic that we employ is to increase the subscription whenever possible, provided we can receive on the layers lacking sufficient buffering at twice the required rate b_l^τ. This choice is motivated by the observation that devoting additional bandwidth to a layer results in diminishing returns since we can only work-ahead and receive the later segments, and the additional bandwidth cannot speed up the process of receiving segments that are imminently required. Experiments suggest that this heuristic of receiving at twice the required rate gives satisfactory performance.

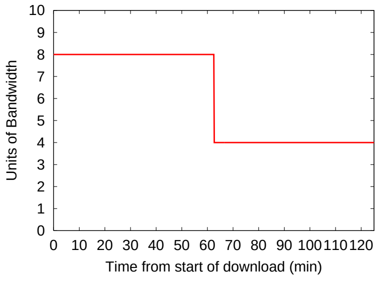
When bandwidth drops below the requirements of the work-ahead policy, the policy tries several strategies to reduce the amount of bandwidth being used. First, it will



(a) Bandwidth Pattern 1



(b) Bandwidth Pattern 2



(c) Bandwidth Pattern 3

Figure 7: Bandwidth Patterns Used For Quality Adaptation Evaluation

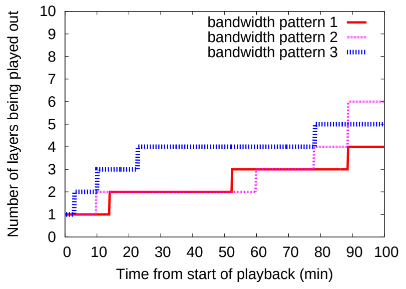


Figure 8: Quality Adaptation with Different Bandwidth Patterns ($s = 8, T = 5$ minutes)

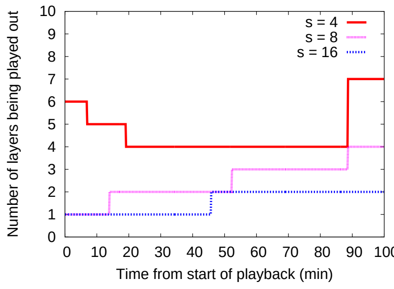


Figure 9: Quality Adaptation with Various Startup Delays ($T =$ 5 minutes, Bandwidth Pattern 1)

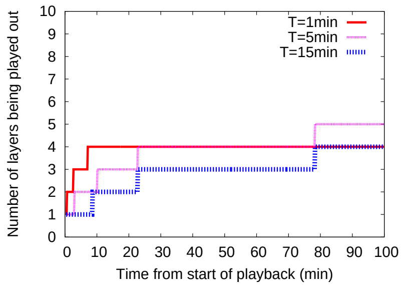


Figure 10: Effect of T on Quality Adaptation ($s = 8$, Bandwidth Pattern 3)

stop work-ahead on layers, starting with the highest quality layers. If this does not reduce bandwidth requirements enough, the policy will then stop downloading on layers, starting with the highest layers. This is because downloading to higher layers increases the risk of receiving data that subsequently cannot be played out if the period of low bandwidth is sustained. If a drop in bandwidth is sustained long enough, layers of quality may need to be dropped according to the following rule: a layer is dropped if one of its segments has not been downloaded in time for playback.

4.4 Performance Evaluation

This section presents sample evaluations of our quality adaptation mechanism assuming a 10 layer media file, where each layer is delivered using OHPB (model 2) with the same parameters, namely $B = 10$, $r = 0.25$, 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 variation is designed to test different aspects of the quality adaptation algorithm. A sawtooth bandwidth variation pattern (Figure 7 (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 7 (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. Since layer addition/removal causes bandwidth usage to change in multiples of $s * r$, the work-ahead policy should show improved bandwidth utilization in conditions of the second bandwidth variation pattern. We also report simulation results that test the performance of the work-ahead policy under exceptional circumstances such as bandwidth decreases halfway through the download of video segments (Figure 7 (c)). The goal was to measure how much quality is preserved in such a case. We have experimented with other bandwidth patterns, but for brevity, restrict our discussion here to the above-mentioned bandwidth patterns.

The performance of the quality adaptation policy under various bandwidth conditions is examined in Figure 8. The protocol is able to exploit bandwidth that is available during the sawtooth and square wave fluctuations. The work-ahead policy uses this extra bandwidth to download segments that occur closer to the end of the media and as a result is able to deliver higher quality at the end of playback. As predicted, the square wave bandwidth pattern enables the protocol to attain higher levels of quality. Since the majority of downloading is done at the beginning of the download, the work-ahead policy is able to sustain very high levels of quality when bandwidth decreases halfway through the download.

Since users may choose to dedicate different amounts of bandwidth to downloading each layer, the work-ahead policy is tested for users who choose to devote, 1, 2 or 4 units of bandwidth per layer. When $r = 0.25$ this is equivalent to users receiving on, $s = 4, 8$ or 16 channels per layer. Figure 9 examines the effectiveness of the work-ahead policy with bandwidth pattern 1 for users who dedicate different amounts of bandwidth for downloading each layer. The work-ahead policy is able to increase playback quality for all of these users. Note that, from Figure 7 (a), the minimum achievable client reception rate over the duration of the download is 4 units of bandwidth.

When the user dedicates 1 unit of bandwidth per layer, it is expected that the user should be able to receive at least 4 layers without the work-ahead policy. With the work-ahead policy the client is able to receive higher levels of quality at the beginning of the download. This can be attributed to the client performing work-ahead before playback begins. However, once playback begins, the client cannot download segments on time for such a high number of layers. Despite not being able to keep up with downloading more layers of quality during playback, the client is still able to use the bandwidth available during fluctuations to download the last segment of the media, completing its playback with 7 layers of quality.

When $s = 8$ the expected number of layers a client should be able to receive without work-ahead is 2. Figure 9 shows that the work-ahead performed when extra bandwidth is available allows the client to receive more than 2 layers for almost half of the playback time. Similarly, the work-ahead policy enables the user with $s = 16$ to receive 2 layers for over half the playback time, when the expected number of layers would be 1.

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 10, for $T = 1$ minute, $T = 5$ minutes, and $T = 15$ minutes. The third bandwidth pattern is chosen because the T values are designed to ensure that the protocol can sustain playback quality when bandwidth drops for extended periods of time. In each case, the percent of data wasted is also measured and it is found that data is only wasted when $T = 1$ minute; in this case, 0.69% of the 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 layers of quality. In these evaluations, $T = 5$ minutes performs the best, providing a balance between adding quality and maintaining work-ahead.

5. CONCLUSIONS

This paper addressed the challenge of streaming popular media files, on-demand, to heterogeneous clients. We considered both heterogeneity in the average available bandwidth and heterogeneity due to time-varying available bandwidth. The new Optimized Heterogeneous Periodic Broadcast (OHPB) we develop 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 startup delays. A novel methodology, based on linear programming models, is used to develop the OHPB segment size progression. For delivery of layered media files using OHPB, we developed efficient quality adaptation mechanisms and policies that allow each client to independently determine how to allocate time-varying available bandwidth among layers at any instant of time such that uniform playback quality can be maintained.

6. ACKNOWLEDGMENTS

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. The authors thank the anonymous ACM Multimedia reviewers for comments that improved the paper presentation.

7. REFERENCES

[1] C. Aggrawal, J. Wolf, and P. Yu. A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *Proc. IEEE ICMCS*, Hiroshima, Japan, June 1996.

[2] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proc. ACM SIGCOMM*, Vancouver, USA, September 1998.

[3] S. Cheung, M. Ammar, and X. Li. On the use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution. In *Proc. IEEE INFOCOM*, San Francisco, USA, March 1996.

[4] D. Eager and M. Vernon. Dynamic Skyscraper Broadcasts for Video-on-Demand. In *Proc. MIS*, Istambul, Turkey, September 1998.

[5] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In *Proc. MMCN*, San Jose, USA, January 2000.

[6] 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.

[7] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Trans. on Networking*, 7(4):458–472, August 1999.

[8] L. Gao, J. Kurose, and D. Towsley. Efficient Schemes for Broadcasting Popular Videos. In *Proc. ACM NOSSDAV*, Cambridge, UK, July 1998.

[9] A. Hu. Video-on-Demand Broadcasting Protocols: A Comprehensive Study. In *Proc. IEEE INFOCOM*, Anchorage, USA, April 2001.

[10] A. Hu, I. Nikolaidis, and P. Beek. On the Design of Efficient Video-on-Demand Broadcast Schemes. In *Proc. MASCOTS*, Maryland, USA, October 1999.

[11] K. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *Proc. ACM SIGCOMM*, Cannes, France, September 1997.

[12] K. A. Hua, O. Bagouet, and D. Oger. Periodic Broadcast Protocol for Heterogeneous Receivers. In *Proc. MMCN*, Santa Clara, USA, January 2003.

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

[14] T. Kim and M. Ammar. A Comparison of Layering and Stream Replication Video Multicast Schemes. In *Proc. ACM NOSSDAV*, Port Jefferson, USA, June 2001.

[15] A. Legout and E. Biersack. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In *Proc. ACM SIGMETRICS*, Santa Clara, USA, June 2000.

[16] X. Li and M. H. Ammar. Bandwidth Control for Replicated-Stream Multicast Video Distribution. In *Proc. HPDC*, Syracuse, USA, August 1996.

[17] X. Li, M. H. Ammar, and S. Paul. Video Multicast over the Internet. *IEEE Network*, 13(2):46–60, April 1999.

[18] M. Luby, V. Goyal, S. Skaria, and G. Horn. Wave and Equation Based Rate Control Using Multicast Round Trip Time. In *Proc. ACM SIGCOMM*, Pittsburgh, USA, September 2002.

[19] A. Mahanti, D. Eager, and M. Vernon. Improving Multirate Congestion Control Using a TCP Vegas Throughput Model. *Computer Networks*, 48(2):113–136, June 2005.

[20] 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.

[21] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proc. ACM SIGCOMM*, Stanford, USA, September 1996.

[22] J. Paris, S. Carter, and D. Long. Efficient Broadcasting Protocols for Video-on-Demand. In *Proc. MASCOTS*, Montreal, Canada, July 1998.

[23] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proc. ACM SIGCOMM*, Cambridge, USA, September 1999.

[24] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-End Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. IEEE INFOCOM*, New York, USA, March 1999.

[25] H. D. Sherali and G. Choi. Recovery of Primal Solutions When Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs. *Operations Research Letters*, 19,

1996.

[26] M. Tantaoui, K. Hua, and T. Do. BroadCatch: A Periodic Broadcast Technique for Heterogeneous Video-on-Demand. *IEEE Trans. on Broadcasting*, 50(3):289–301, 2004.

[27] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like Congestion Control for Layered Video Multicast Data Transfer. In *Proc. IEEE INFOCOM*, San Francisco, USA, April 1998.

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

[29] Y. Zhao, D. Eager, and M. Vernon. Network Bandwidth Requirements for Scalable On-Demand Streaming. In *Proc. IEEE INFOCOM*, New York, USA, June 2002.

[30] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. Subjective Impression of Variations in Layer Encoded Videos. In *Proc. IEEE IWQoS*, Monterey, USA, June 2003.

APPENDIX

A. SUBGRADIENT ALGORITHM FOR THE OHPB LINEAR PROGRAM

To construct the subgradient algorithm, we first reformulate the OHPB LP as follows. Observe that constraint groups (3)-(6) essentially impose inequality relations between τ_j and linear combinations of l_i. By successive substitutions of (4)-(6) into (3), we obtain the revised OHPB LP, which is equivalent to the original:

Minimize $\sum_j w_j \tau_j$

Subject to:

$$\begin{cases} \sum_i l_i = L \\ \tau_1 \geq \alpha_{11}^{(1)} l_1 + \alpha_{11}^{(2)} + \ldots + \alpha_{11}^{(K)} \\ \cdots \\ \tau_1 \geq \alpha_{1K}^{(1)} l_1 + \alpha_{1K}^{(2)} + \ldots + \alpha_{1K}^{(K)} \\ \cdots \\ \tau_m \geq \alpha_{m1}^{(1)} l_1 + \alpha_{m1}^{(2)} + \ldots + \alpha_{m1}^{(K)} \\ \cdots \\ \tau_m \geq \alpha_{mK}^{(1)} l_1 + \alpha_{mK}^{(2)} + \ldots + \alpha_{mK}^{(K)} \end{cases}$$

$$l_i, \tau_j \geq 0, \qquad \forall i, j$$

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

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:

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

$$= \max_{\lambda_{ij} \geq 0} \min_{l \in \mathcal{P}_1} [\sum_i (w_i - \sum_j \lambda_{ij}) \tau_i + \sum_i \sum_j (\lambda_{ij} \sum_k (\alpha_{ij}^{(k)} l_i))] \qquad (A.1)$$

Note that dual feasibility requires $\sum_j \lambda_{ij} \leq w_i$, because otherwise the inner minimization is unbounded. Therefore, (A.1) is equivalent to:

$$\max_{\lambda \in \mathcal{P}_2} \min_{l \in \mathcal{P}_1} \sum_i \sum_j (\lambda_{ij} \sum_k (\alpha_{ij}^{(k)} l_i)) \qquad (A.2)$$

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 (A.2) 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:

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

The minimization problem above is reformulated into:

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

Let $i^* = \text{argmin}_i \beta_j$, 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:

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

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 [25].