Optimal Delay for Media-on-Demand with Pre-loading and Pre-buffering

Amotz Bar-Noy * Richard E. Ladner † Tami Tamir ‡

February 16, 2006

Abstract

Broadcasting popular media to clients is the ultimate scalable solution for media-on-demand. The simple solution of downloading and viewing the media from one channel cannot guarantee a reasonable startup delay for viewing with no interruptions. Two known techniques to reduce the delay are pre-loading and pre-buffering. In the former an initial segment of the media is already in the client buffer, and in the latter segments of the media are not transmitted in sequence and clients may pre-buffer later segments of the media before viewing them. In both techniques, the client should be capable to receive streams from channels at the same time of handling its own buffer and view the media from either one of the channels or the buffer.

In this paper we consider broadcasting schemes that combine pre-loading and pre-buffering. We present a complete tradeoff between (i) the size of the pre-loading; (ii) the maximal possible delay for an uninterrupted playback; (iii) the number of media; and (iv) the number of channels allocated per one media. For a given B the size of the pre-loading as a fraction of the media length, for m media, and for h channels per media, we first establish a lower bound for the minimal maximum delay, D, as a fraction of the movie length, for an uninterrupted playback of any media out of the m media. We then present an upper bound that approaches this lower bound when each media can be fragmented to many segments.

*Computer and Information Science Department, Brooklyn College, 2900 Bedford Avenue Brooklyn, NY 11210. E-mail: *amotz@sci.brooklyn.cuny.edu*.

†Department of Computer Science and Engineering, Box 352350, University of Washington, Seattle, WA 98195. This work was partially supported by NSF grant No. CCR-0098012. E-mail: *ladner@cs.washington.edu*.

‡School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail: *tami@idc.ac.il*.

1 Introduction

Media-on-demand (MoD) is the demand by clients to read, listen, or view various types of media. In its simplest function, the clients would like to have an uninterrupted playback with as minimal as possible start-up delay. The subject of this paper is to reduce the maximal start-up delay for MoD systems that support uninterrupted service. Our main objective is to achieve the smallest maximal start-up delay for given amount of two resources: the system's bandwidth, and the client local memory.

There are two main types of systems that support MoD: unicast systems and broadcast systems. The former guarantees an immediate service as long as there are not too many clients. The latter can support many clients but cannot guarantee immediate service. For popular media, *broadcasting* is the ultimate scalable solution. In various broadcasting schemes, different parts of the media are transmitted on channels viewable to the clients. This paper considers the potential benefit of broadcasting schemes from using some of the client memory for storing in advance (pre-loading) parts of the media.

In the simplest implementation of MoD systems, clients who wish to view a movie[1], select the channel that would start broadcasting this movie the earliest after their request time. Movies are broadcast on one channel or several channels. Thus if h channels are allocated to a movie of length L time units, the maximal start-up delay is L/h units by starting a new transmission every L/h time units.

In recent years, more efficient broadcasting schemes that are based on pre-buffering were suggested. In these schemes, each movie is partitioned into segments, and the segments are transmitted on the channels in some order, not necessarily their order in the movie. The client is reading all the channels simultaneously, 'collecting' segments to its local memory, and watch the segments of the movie in order - some directly from the channels and some from its own memory.

The above broadcasting schemes require customers to get the service through a set-top-box (STB) capable of storing locally the transmitted data. This requires that the STB will be equipped with a local memory (disk). In fact, this technology is already available: digital VCRs offered by ReplayTV [19], TiVo [22], and UltimateTV [23], have capacities of at least 300 gigabytes, enabling the client to store hours of movies in perfect quality. The disk capacity can be used to store entire movies and also pre-buffered segments and pre-loaded segments of other movies. Usually the former type of movies will be non-popular movies where the latter type will be popular movies for which the broadcasting solution is more beneficial. In this paper, we consider broadcasting schemes that combine pre-loading and pre-buffering. That is, we assume that some prefix of the movie is stored at the client's machine, and therefore he or she should only receive the reminder of the movie. We present a complete tradeoff between (i) the size of the pre-loading; (ii) the maximal possible delay for an uninterrupted playback; (iii) the number of movies; and (iv) the number of channels allocated per one movie.

For a given B the size of the pre-loading as a fraction of the movie length, for m movies,

[1]For convenience, we use the terminology of movies in Video-on-Demand (VoD).

and for h channels per movie, we first establish a lower bound for the minimal maximum delay, D, as a fraction of the movie length, for an uninterrupted playback of any movie out of the m movies. We then present an upper bound that approaches this lower bound when each movie can be fragmented to many segments.

1.1 Model and Preliminaries

The system broadcasts m movies on h channels. Unless specified otherwise, assume that all m movies have the same length, L, normalized to be one time unit ($L = 1$). Each movie is partitioned into s segments of equal length. Segment size may range from a single bit (which is theoretically interesting) to the whole movie (in case of a single segment). The segments are indexed 1 to s in the order they should be viewed. The segments of the movies may be broadcast in any order on any channel. Assume that it takes one time slot to transmit or view a segment and thus, the length of the time slot is $1/s$. Assume further that all the channels are synchronized in the sense that the starting points for the time slots coincide in all of them. Clients may buffer or view segments from any channel since they may receive data from all of them. In other words, the receiving bandwidth of each client is h. This implies that a client buffer or view segment i the first time he or she can do so after their arrival time. Clients may buffer any number of segments before the viewing process begins.

The maximal possible delay of a client is denoted by D and is given as a fraction of the movie length. That is, if for example $D = 1/4$, no client will wait more than $1/4$ of a movie length till it can start an uninterrupted playback of the movie. Let $d = Ds$ denote the maximal delay measured as number of segments (time-slots). In the broadcasting schemes we present, the maximum delay is given in units of time-slots, thus we assume that D is a multiple of $1/s$.

The basic principle in all the schemes that use pre-buffering is that early segments should be broadcast more frequently than later segments. Intuitively, a client needs to watch the z^{th} segment only $z - 1$ time-slots after it starts watching the movie, therefore, the z^{th} segment, can be transmitted less often than earlier segments. Formally, in [2], optimal schemes that are based on pre-buffering are developed using the *windows scheduling* problem and the following is shown:

Theorem 1.1 *Let $\mathcal{S}$ be a schedule that broadcasts $s \geq 1$ segments for a movie on $h \geq 1$ channels. Then $\mathcal{S}$ guarantees a maximum start-up delay of $d > 0$ time-slots if and only if segment z is transmitted once in any window of $d + z - 1$ segments for each $1 \leq z \leq s$.*

Assume now that out of the s segments composing the movie, the first b are *pre-loaded* and are stored at the client's local machine (set top box), the other $s - b$ segments are transmitted on channels. Clearly, the client can always watch the first b segments with no delay. Consider the remainder of the movie as a complete (shorter) movie. Assume there exists a broadcasting scheme that enables any client to view this movie with delay at most d' (in number of segments units). The idea is to overlap the time the client watches the first b segments with the time he or she is waiting to the rest of the data. This would result in a delay $\max(0, d' - b)$. The challenge is to schedule the remaining $s - b$ segments on the broadcasting channels in a way that minimizes this term.

Example: Consider a single movie transmitted on a single channel. Assume that the client has at his local machine all but the last 5 segments, which are not pre-loaded, and are transmitted on the channel in the following (repeated) order:

$$[1, 3, 2, 4, 1, 5, 2, 3, 1, 4, 2, 5]$$

In this order, the segments $1, 2$ are transmitted every 4 slots and the segments $3, 4, 5$ are transmitted every 6 slots. Recall that the movie is partitioned into s segments, thus, these 5 segments are segments $s-4, \ldots, s$ of the movie. The first $b = s - 5$ segments are pre-loaded and available to the client at any time (thus, $B = (s-5)/s$). By Theorem 1.1 the above transmission of the last 5 segments guarantees a delay of at most $d' = 4$ slots for viewing with no interruptions the last 5 segments. Thus, if $s \geq 9$, or equivalently, $b \geq 4$ meaning that the client has at least the first 4 segments of the movie, then there is no delay at all. If $s < 9$, the delay with pre-loading is $4 - b$ slots which is $D = (4-b)/(b+5) = (9-s)/s$ of the whole movie. We get the following tradeoff between B and D:

s	B	D
5	0	4/5
6	1/6	3/6
7	2/7	2/7
8	3/8	1/8
9	4/9	0
> 9	$(s-5)/s$	0

In particular, this means that in order to guarantee no delay the pre-loading size should be 4/9 of the movie length, and with no pre-loading the maximal delay is 4/5 of the movie length.

Table 1 provides a glossary of the notation used in the paper.

notation	meaning
h	number of channels
m	number of different movie
$\rho = h/m$	the ratio between number of channels and number of movies.
s	number of segments per each movie.
B	the size of the pre-loading buffer as a fraction of the movie length.
$b = Bs$	the size of the pre-loading buffer as a number of segments
D	the maximal delay for an uninterrupted playback as a fraction of the movie length.
$d = Ds$	the maximal delay as a number of segments
d'	the maximal delay for the non pre-loaded part, as a number of segments

Table 1: Glossary of notations.

The lower bound and the matching broadcasting scheme we present assume that the client's memory stores only prefixes of movies. One might doubt that this is optimal, and suggest it might be better to store late parts of the movie and broadcast earlier ones. The following

Theorem should remove such doubts - it states that the best way to use an allocated amount B of memory to a movie is by storing (pre-loading) a prefix of size B of this movie.

Theorem 1.2 *For any broadcasting scheme that combines pre-loading and broadcasting, if memory of size B is allocated to a movie, then it is optimal to store from this movie a* prefix *of size B.*

Proof: The idea is similar to the optimality proof of the offline algorithm for caching - that evicts from the cache the page that will be requested last among the pages that are currently in the cache. Consider any broadcasting scheme S in which for some movie there exists a bit i that is not pre-loaded, while some bit $j > i$ is. Since j is pre-loaded, it is never transmitted by the scheme.

Consider the scheme S' in which bit i is pre-loaded and bit j is transmitted whenever bit i was transmitted in S. Clearly, the user will have bit i on time (from its memory) and bit j will be available at the time bit i was available in S. Since we assume that all clients read the movie in order, bit i is requested before bit j, therefore, by having bit j available in S' at the time i was available in S, the client's delay can only decrease. □

1.2 Related Work

MoD systems, and in particular the solution of broadcasting, have been studied extensively in recent years. The paper [3] surveys broadcasting protocols and describes the development of these protocols, starting with Staggered broadcasting protocols, in which the movies are simply transmitted repeatedly on the channels (e.g., [4]), through Pyramid-based broadcasting protocols, in which movies are partitioned into segments and different segments are broadcast on different channels [24], and finally Harmonic broadcasting protocols in which segment i is allocated bandwidth proportional to $1/i$ (e.g., [8]).

The case when there is no pre-loading and pre-buffering may start only when clients start viewing the movie received much attention in the recent decade. The papers [7, 9] present a simple schedule of one movie on h channels by partitioning the movie into $2^h - 1$ segments. Their schedule implies a maximal start-up delay of $1/(2^h - 1)$ for a movie of length 1. This scheme is improved in the Pagoda scheme ([15]), the new Pagoda scheme ([11]), the Recursive Frequency-Splitting scheme ([21]), the Harmonic broadcasting scheme ([1])), and the Polyharmonic broadcasting scheme ([14]. In these schemes, the worst-case maximal delay asymptotically approaches $1/(e^b - 1)$ for total bandwidth b. Several papers, e.g., [6] have shown this bound on delay to be optimal.

Harmonic broadcasting is implemented in [1] by a reduction from the *window-scheduling* problem. Specifically, the movie is partitioned into s equal-sized segments that are scheduled on the channels such that the gap between any two consecutive appearances of segment i is at most i. For a given number of channels, the goal is to maximize s, and as a result, minimize the start-up delay (which is at most $1/s$). A schedule based on this principle is shown to approach the lower bound as $h \to \infty$. The papers [12, 13] also allow clients to start buffering segments before they start viewing the movie to achieve better results. However, they demonstrate the

usefulness of this observation only for small examples. The paper [2] gives asymptotic matching upper and lower bound on the maximal delay of a broadcasting scheme that uses pre-buffering.

The papers [18, 16] consider pre-loading, but only for the case of zero delay. The paper [18] does not allow pre-buffering before the clients start watching the movie whereas the paper [16] improve the results by allowing this feature. In another work on pre-loading [10], it is assumed that each client pre-loads segments of a different set of movies, according to the client's choice. Earlier work on pre-loading assume that the preloaded data is stored at a proxy server and not at the client's local machine [5, 20].

1.3 Contribution

We consider broadcasting schemes that combine pre-loading and pre-buffering. We present a complete tradeoff between (i) the size of the pre-loading; (ii) the maximal possible delay for an uninterrupted playback; (iii) the number of media; and (iv) the number of channels allocated per one media.

For a given B the size of the pre-loading as a fraction of the media length, for m media, and for h channels per media, we first establish a lower bound for the minimal maximum delay, D, as a fraction of the movie length, for an uninterrupted playback of any media out of the m media. We then present an upper bound that approaches this lower bound when each media can be fragmented to many segments.

2 A Lower Bound for the Maximal Delay

We first compute a lower bound for the maximal delay for a fix $s \geq 1$ number of segments per movie. Then we calculate the general lower bound by letting s tend to infinity. For ease of presentation we assume that both $b = Bs$ and $d = Ds$ are integers.

Each client has the first $b = Bs$ segments of each movie in its buffer. Therefore, the channels needs to broadcast only segments $b+1, \ldots, s$. Since the maximal delay is d, segment i of each movie should be broadcast at least once in any window of size $d+i$ for $b+1 \leq i \leq s$. That is, segment i consumes at least $1/(d+i)$ fraction of a channel. Since the total number of channels is h and since there are m movie, it follows that

$$m \sum_{i=b+1}^{s} \frac{1}{i+d} \leq h \ .$$

This is equivalent to

$$\sum_{i=b+d+1}^{s+d} \frac{1}{i} \leq \rho \ .$$

Using the known bound on the harmonic number $H_n = \sum_{i=1}^{n}(1/i)$ implies

$$\ln\left(\frac{s+d}{b+d}\right) \leq \rho \ .$$

Since $b = Bs$ and $d = Ds$, this is equivalent to

$$\frac{1+D}{B+D} \leq e^{\rho} \ .$$

By manipulating the above inequality we get the lower bound for D given B

$$D \geq \frac{1 - Be^{\rho}}{e^{\rho} - 1} \ .$$

Equivalently, the lower bound for B given D is

$$B \geq \frac{1 - D(e^{\rho} - 1)}{e^{\rho}} \ .$$

In particular, when $B = 0$ the lower bound matches the known lower bound

$$D \geq \frac{1}{e^{\rho} - 1} \ .$$

When $D = 0$ the lower bound for B is

$$B \geq \frac{1}{e^{\rho}} \ .$$

For example, in order to guarantee no delay for a single movie transmitted on a single channel the client must pre-load at least $1/e \approx 0.368$ of the movie.

3 Optimal Schedules

Assume first that $m = 1$. Consider a schedule of the numbers $[x..y]$ on h channels such that for any $x \leq i \leq y$, in each window of i consecutive slots the number i appears at least once in one of the channels. For example

$$[4, 6, 5, 7, 4, 8, 5, 6, 4, 7, 5, 8]$$

is such a schedule for $h = 1$, $x = 4$, and $y = 8$.

Suppose we interpret the numbers $x, \ldots, y$ as segments $s - y + x, \ldots, s$ of the movie. This reflects a partition of the movie into s segments each of length $1/s$ of the movie length. Moreover, the pre-loading size should be $b = s - y + x - 1$ which implies $B = (s - y + x - 1)/s$. Furthermore, the delay with pre-buffering is $D = (y + 1 - s)/s$. It follows that a viable range for s is from $y - x + 1$ to $y + 1$ (it might be that $s > y + 1$ but the delay never reduces below 0), and we get the following tradeoff between B and D:

s	B	D
$y - x + 1$	0	$x/(y - x + 1)$
$y - x + 2$	$1/(y - x + 2)$	$(x - 1)/(y - x + 2)$
$y - x + i$	$(i - 1)/(y - x + i)$	$(x - i + 1)/(y - x + i)$
y	$(x - 1)/y$	$1/y$
$y + 1$	$x/(y + 1)$	0
$> y + 1$	$(s + x - y - 1)/s$	0

In particular, this means that in order to guarantee no delay with this schedule the pre-loading size should be $x/(y+1)$ of the movie length and with no pre-loading the maximal delay is $x/(y-x+1)$ of the movie length.

Consider the case $s = y-x+i$ in which $B = (i-1)/(y-x+i)$ and $D = (x-i+1)/(y-x+i)$. Assign $z = y+1$ and $j = x-i+1$. With these variables,

$$B = \frac{x-j}{z-j} \qquad D = \frac{j}{z-j} \ .$$

Further, assign $w = z/j$. It follows that

$$B = \frac{x/j-1}{w-1} \qquad D = \frac{1}{w-1} \ .$$

As shown in [2], in the limit there exists a schedule $[x..y]$ such that

$$\frac{1}{e^h-1} \approx \frac{x}{y-x+1} \ .$$

This implies that

$$x = \frac{y+1}{e^h} = \frac{z}{e^h} \ .$$

Furthermore, the values of B and D as a function of w are

$$B = \frac{w/e^h-1}{w-1} \qquad D = \frac{1}{w-1} \ .$$

Plugging $w = 1+1/D$ in the equality for B yields

$$B = D\left(\frac{1+1/D}{e^h}-1\right) = \frac{D+1}{e^h} - D = \frac{1}{e^h} - \left(1-\frac{1}{e^h}\right)D \ .$$

Equivalently,

$$D = \frac{1/e^h - B}{1-1/e^h} = \frac{1-e^hB}{e^h-1} \ .$$

For the special case of $D = 0$ we have $(1-e^hB) = 0$ or equivalently

$$B = \frac{1}{e^h} \ .$$

For the special case of $B = 0$ we have $1/e^h = (1-1/e^h)D$ or equivalently

$$D = \frac{1}{e^h-1} \ .$$

The calculation for the general case of $m > 1$ is identical. For each of the m movies, segments $s-y+x, \ldots, s$ are transmitted with windows $x, \ldots, y$, respectively. Along the whole calculation it is possible to replace h by $\rho = h/m$. Note that all the above upper bounds match the lower bounds from Section 2

4 Discussion

In this paper we showed a tradeoff between the size of the pre-loaded buffer and the guaranteed delay for an uninterrupted playback of movies. We first proved the optimal possible tradeoff and then demonstrated how to achieve it when a movie may be partitioned to many segments. In what follows we discuss several possible extensions.

Limiting the receiving bandwidth: In this paper we assumed that a client can buffer segments of the movie from all the channels. This means that the receiving bandwidth of a client is h times more than the playback bandwidth. Several papers explored the case where the receiving bandwidth is only r times the playback bandwidth for some $1 < r < h$ (e.g. [17]). However, no paper consider this case with the pre-loading capability.

Limited size buffers: Early works on this model assumed that the buffer size for the pre-buffered segments is bounded as a fraction of the movie length (see the survey [3]). Although it seems that the sky is the limit for cheap and large memory, this might not be the case for hand-held set top boxes. It is interesting therefore to investigate the tradeoff between the pre-loaded buffer size and the pre-buffered buffer size when their sum is bounded.

Movies with different popularity: The solution of broadcasting (in contrast to unicast) is suitable for popular media. However, even among popular media there are different levels of popularity. In particular, only a small number of movies is very popular at a specific time. It is very intriguing to see how the combination of pre-loading and pre-buffering can be used to provide smaller delay to the highly requested movies while increasing the maximal possible delay for less popular movies. The problem can be modelled as follows. Consider a system with m movies with different popularity. The popularity parameter of movie i is denoted p_i such that $\sum_{i=1}^{m} \frac{1}{p_i} = 1$. The parameter p_i can be viewed as the probability that the next client's request is to watch movie i. Let D_i denote the maximal possible delay a broadcasting scheme guarantees for a movie i, then the goal is to minimize $\sum_{i=1}^{m} p_i D_i$. That is, the weighted maximal possible delay (also the expected maximal delay) of the whole system. In practice, especially since the popularity parameter varies drastically with the time, it is not practical to assume that each movie has a specific popularity parameter and instead a simpler model may be addressed. The system distinguishes between the *hot* movies and the rest of the popular movies. There are various ways to ensure smaller delay to the hot movies, they can be transmitted more often, or a larger portion of these movies might be pre-loaded.

References

[1] A. Bar-Noy and R. E. Ladner. Windows Scheduling Problems for Broadcast Systems. *SIAM Journal on Computing (SICOMP),* 32(4):1091–1113, 2003.

[2] A. Bar-Noy, R. E. Ladner, and T. Tamir. Scheduling techniques for media-on-demand. *Proc. of the 14-th Annual ACM-SIAM Symposium on Discrete Algorithms,* 791-800, 2003.

[3] S. W. Carter, D. D. E. Long, and J. Pâris. Video-on-Demand Broadcasting Protocols. In *Multimedia Communications: Directions and Innovations (J. D. Gibson, Editors),* Academic Press, San Diego, 179–189, 2000.

[4] A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic Batching Policies for an On-Demand Video Server. *ACM Multimedia Systems Journal,* 4(3):112–121, 1996.

[5] D. Eager, M. Ferris and M. Vernon. Optimized regional caching for on-demand data delivery. In *Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99)*, 1999.

[6] L. Engebretsen and M. Sudan. Harmonic Broadcasting is Optimal. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA),* 431–432, 2002.

[7] K. A. Hua, Y. Cai, and S. Sheu Exploiting Client Bandwidth for More Efficient Video Broadcast. In *Proceedings of the 7th International Conference on Computer Communication and Networks (ICCCN),* 848–856, 1998.

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

[9] L. Juhn and L. Tseng. Fast Data Broadcasting and Receiving Scheme for Popular Video Service. *IEEE Transactions on Broadcasting,* 44(1):100–105, 1998.

[10] J. F. Pâris. A Broadcasting Protocol for Video-on-Demand Using Optional Partial Preloading. In *Proceedings of the 11th International Conference on Computing*, vol.I, 319-329, 2002.

[11] J. Pâris. A Simple Low-Bandwidth Broadcasting Protocol for Video-on-Demand. In *Proceedings of the 8th International Conference on Computer Communications and Networks (IC3N),* 118–123, 1999.

[12] J. Pâris. A Fixed-Delay Broadcasting Protocol for Video-on-Demand. In *Proceedings of the 10th International Conference on Computer Communications and Networks (IC3N),* 418–423, 2001.

[13] J. Pâris. A Simple but Efficient Broadcasting Protocol for Video-on-Demand. In *Proceedings of the 24th International Performance of Computers and Communication Conference (IPCCC 2005),* 167-174, 2005.

[14] J. Pâris, S. W. Carter, and D. D. E. Long. A Low Bandwidth Broadcasting Protocol for Video on Demand. In *Proceedings of the 7th International Conference on Computer Communications and Networks (IC3N),* 690–697, 1998.

[15] J. Pâris, S. W. Carter, and D. D. E. Long. A Hybrid Broadcasting Protocol for Video on Demand. In *Proceedings of the IS&T/SPIE Conference on Multimedia Computing and Networking (MMCN),* 317–326, 1999.

[16] J. Pâris and D. D. E. Long. The Case for Aggressive Partial Preloading in Broadcasting Protocols for Video-on-Demand. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 113–116, 2001.

[17] J. Pâris and D. D. E. Long. Limiting the Receiving Bandwidth of Broadcasting Protocols for Video-on-Demand. In *Proceedings of the Euromedia Conference*, 107-111, 2000.

[18] J. Pâris, D. D. E. Long, and P. E. Mantey, Zero-Delay Broadcasting Protocols for Video-on-Demand. In *Proceedings of the 1999 ACM Multimedia Conference* pages 189-197,

[19] ReplayTV. *http://www.replay.com*

[20] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of the IEEE 18th Conference on Computer Communications (INFOCOM)*, 1310–1319, 1999.

[21] Y. C. Tseng, M. H. Yang, and C. H. Chang. A Recursive Frequency-Splitting Scheme for Broadcasting Hot Video in VOD Service. *IEEE Transactions on Communications,* 50(8):1348–1355, 2002.

[22] TiVo Technologies. *http://www.tivo.com*

[23] UltimateTV. *http://www.ultimatetv.com*

[24] S. Viswanathan and T. Imielinski. Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting. *ACM Multimedia Systems,* 4(3):197–208, 1996.